Large-scale Structured Learning

Siddharth Gopal

August 2014

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Yiming Yang (Chair)
Jaime Carbonell
Andrew Moore
Thomas Hofmann (External)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Contents

Abstract

In this thesis we study large-scale structured learning in the context of supervised, unsupervised and semi-supervised learning. In the big data era, it is increasingly important to automatically infer structure from the data or leverage human provided structures in various learning processes.

In the first part of this thesis, we focus on how to harness external supervision about the structural dependencies between class-labels to improve classification performance. Specifically we develop models that can exploit hierarchical and graph based dependencies between class-labels, in a tractable manner. We propose two frameworks in this context (a) A Hierarchical Bayesian model that can leverage hierarchical dependencies using Bayesian priors (b) A non-Bayesian Risk minimization framework that can incorporate hierarchical and graphical dependencies into the regularization structure of the model parameters. For both frameworks we develop fast inference and training methods that can easily handle large-scale structures with hundreds of thousands of classes. We also develop parallel iterative training procedures for well-studied conditional models for flat multiclassification (and other tasks) that enhances the scalability in the presence of large number of outcomes.

In the second part of this thesis, we focus on cases where human-generated structures for organizing data are not readily available and needs to be generated. Using the von Mises-Fisher distribution as the building block we propose three increasingly powerful Bayesian models that can recover flat, hierarchical as well as temporal structures from data on unit-sphere manifolds. Our experiments on multiple datasets showed that our proposed models are better alternatives to existing topic models based on multinomial and Gaussian distributions in their ability to recover ground-truth clusters.

In the third part of the thesis, the semisupervised setting, we focus on how to expand a set of human-specificied classes (or clusters) in a manner that is consistent with user expectations. We propose two new frameworks in this regard, a probabilistic framework and a constrained optimization framework. Both of them rely on learning a transformation of the instances such that the clusters in the transformed space match user expectations better than the original space. Our extensive evaluation on several application domains showed significant improvement in clustering performance over other competing methods.

1 Introduction

The availability of massive amounts of data in digital form has fueled the need to provide a structured and organized view of the same for effective search, browsing and data mining. The growing size of various user content on the internet (blogposts, pictures, text, videos), the increasing collection of digitized books and consumer behaviour for analytics, petabytes of astronomical observations every day are few examples of steadily growing data. The goal of the thesis is to develop models and algorithms to assist the user in providing a structured and easily accessible organization of such large-scale data. The thesis is broadly divided into three parts corresponding to structured learning in the context of supervised, unsupervised and semisupervised settings.

1.1 Large-scale Classification with Structures

Structuring the data is one of the first and key steps that is done to make the data usable for any form of data analysis, exploration or search. For instance, a hierarchical structure (or a taxonomy) provides a natural and convenient way to organize large amounts of data and has widespread usage in several domains; the Yahoo! directory taxonomy[1] for *webpages*, the International patent classification hierarchy for easy browsing/retrieval of *patents*, the extensive *shopping product* hierarchy of Amazon [2] and Google [3], the Medical Subject Heading hierarchy for indexing millions of research articles in PubMed are excellent examples of human generated structures to organize data in important applications. Another commonly used structure for

[1] http://dir.yahoo.com/
[2] http://www.amazon.com/gp/site-directory
[3] https://support.google.com/merchants/answer/160081?hl=en

organizing data is a graph, which enables easy traversal to related data points - the most popular example is the Wikipedia graph that connects related Wikipedia categories with each other.

Once such large-scale structures have been created, it is naturally desirable to fold in new incoming data into one or more nodes (classes) in this structure. This challenging problem will be the focus of the first part of this thesis, where we develop classifiers that can be efficiently trained to classify incoming data into one or more classes. Unlike typical classification systems, the hierarchical or the graphical structures associated with the class-labels presents several interesting challenges:

1. *Label depedencies*: Structures can provide valuable information while trying to classify incoming data. For example, consider the Yahoo! Directory for webpages, based on the hierarchical structure between the class-labels one can say that a webpage that belongs to class 'Disease' is also likely to be a member of class 'Health' but unlikely to be a member of 'Music'. One-against-rest classifiers, while most popular due to their simplicity, cannot model such structural dependencies between the class-labels because classifiers are trained independently and hence lack the necessary expressive power.

2. *Data-sparsity*: One of the common problems in large-scale hierarchies is that majority of the classes in the hierarchy have only a few positive training examples. This highly skewed distribution of training examples across the classes is a well documented problem in hierarchical classification literature [Liu et al., 2005], [Yang et al., 2003], [Bennett and Nguyen, 2009]. For example, 76% of class-labels in the Yahoo! Directory have less than 5 positive instances [Liu et al., 2005] and 63% of class-labels in the English portion of Wikipedia have less than 10 positive instances. Learning independent models (one per class-label) with such limited training examples would be suboptimal due to overfitting. Joint learning would be a better alternative and would enable us to effectively 'borrow' the positive training examples from related classes based on the proximity of classes in the provided hierarchies or graphs.

3. *Scability*: The sheer sizes of very large hierarchies and graphs present a significant computational challenge for classification algorithms. For example, the Wikipedia dataset[4], one of the common benchmark datasets, consists of approximately 2.4 million Wikipedia articles spanning across 325,000 classes (with graphical dependencies) and a vocabulary size of 1.6 million words. Even for simple one-versus-rest classifiers, this amounts to learning approximately half a trillion parameters (325,000 x 1.6 million) which is roughly 2 Terabytes of storage. This is computationally challenging even for training simple multiclass

[4]http://lshtc.iit.demokritos.gr/

(K-way) classifiers and ignoring the dependencies among class-labels. Jointly learning the parameters based on the class-label dependencies is even more challenging.

Addressing these challenges will be the focus of the first part of the thesis in chapters 2, 3 and 4

- In chapter 2 we develop a Bayesian framework for large-scale classification that can exploit hierarchical dependencies between the classes to improve performance. Our model is based on multivariate logistic regression and harnesses the parent-child relationships in the taxonomy by placing a hierarchical prior over the children nodes centered around the parameters of the parent. We develop fast variational inference algorithms and further approximations for large-scale structures which are not only orders of magnitude faster than their typical sampling (MCMC) couterparts but offer much better classification performance than other flat and hierarchical methods.

- In chapter 3 we develop the first large-margin framework that can exploit both hierarchical and graph based dependencies between class-labels to improve classification. Our framework relies on directly incorporating the dependencies between the classes into the regularization structure of model parameters. The simplicity of the model helps us develop fast parallelizable training methods that can easily scale to very large-scale structures - orders of magnitude larger than what existing hierarchical methods have been scaled to.

- In chapter 4, in the absence of additional structural information between classes, we develop a distributed approach to train flat multiclass logistic regression models. Our parallelization strategy is based on splitting the computation of the normalization constant across multiple computing units and enables us to scale to very large number of classes. This is the first work that shows multiclass logistic regression (and in general, conditional models like maxent) can be scaled to tens of thousands of Multinomial outcomes.

1.2 Unsupervised Structure Extraction

In many domains, such human-generated structures for organizing data collections might not always be readily available. In such cases it is highly desirable to provide organized views of the data to help the end users understand the data collection better. For example, consider the citeseer dataset [5]; this dataset consists of 1 million research papers in computer science spanning over a decade. Let us consider a user who wants to know how the field of computer science has progressed in the last decade. It would be hard for the user to go through all

[5] http://csxstatic.ist.psu.edu/about/data

research articles to understand the different subfields of computer science and how each of the subfield evolved over time. It would be much easier if we can generate a browser that brings out the underlying themes in the research articles. For larger corpora especially, it might be even more beneficial to generate a hierarchical browser that enables users to zoom in and zoom out of the different subfields/subsubsfields in computer science, or a temporal browser that shows the rise and decline of the different subfields over time.

There are a wide variety of methods that already cater to such needs, including some of the most popular topic models such as Latent Dirichlet Allocation [Blei et al., 2003] , Hierarchical Topic model [Blei et al., 2004] , Dynamic Topic Models [Blei and Lafferty, 2006a] and their several other variants [Wang and McCallum, 2006], [Wang et al., 2012], [Blei and Lafferty, 2007]. Such probabilistic models define a generative model by assuming some rigid instance representation, for e.g. Multinomial Mixtures assumes that each instance is represented as discrete feature-counts and is drawn from one of many Multinomial distributions.

However, it is questionable whether such representation of data is appropriate for all domains. For example, in text-mining (classification, retrieval, collaborative filtering etc) documents have typically been represented using a term frequency-Inverse Document frequency Normalized form (Tf-Idf normalization) [Salton and McGill, 1986], where each document is represented as a point on a unit-sphere using a combination of both within-document frequencies and inverse corpus frequencies. Tf-Idf normalization has always shown better performance than feature-counts representation based on several supervised tasks such as classification [Joachims, 2002], retrieval [Robertson, 2004] etc. Similarly, in Image-modeling, unit normalized spatial pyramid vectors is a common representation [Yang et al., 2009]. Normalization is often an important step in data analysis because it removes the 'magnitude' of the instances from the picture and places more importance on the directional distribution; in other words, we do not want unduly long documents or big images to distort our inferences, hence we represent the instances as relative contributions of individual features.

Despite the practical successes of normalized data representation, they have not been well studied by Bayesian Graphical models. Since the data lies on a unit-sphere manifolds, popular clustering assumptions such as Gaussian [O'Hagan et al., 2004] or Multinomial [Blei et al., 2003, 2004, Blei and Lafferty, 2006a,b] are not appropriate. On one hand we have empirical success of such normalized representation and on the other hand we have a wide variety of graphical models that model data using a different representation. Can we bridge the gap between the two approaches? Can we develop models that combine the strengths of both by developing new graphical models for the data on unit-sphere manifolds? We address this challenge in chapter 5 where we develop a suite of models for clustering high-dimensional data on a unit sphere based on the von Mises-Fisher (vMF) distribution,

- We define a comprehensive fully Bayesian formulation of a mixture of vMF distributions that enables information sharing among clusters. Our formulation is analagous to a Bayesian mixture of Gaussian distributions or Multinomial distributions but is especially suited for normalized instance representation like Tf-Idf. We develop efficient algorithms for posterior inference based on both variational inference methods as well as collapsed gibbs sampling techniques. Our experiments on several datasets show significant improvement in performance over other popular clustering models such as K-means, Multinomial mixtures and Latent Dirichlet Allocation.
- For large-scale corpora, we extend our fully Bayesian mixture of vMF distributions to generate hierarchically coherent (Hierarchical vMF mixture) or temporally meaningful (Temporal vMF mixture) clusters in the data. In the former we enable partitioning the data into increasing levels of specificity as defined by a given input hierarchy while in the latter we redefine the static mixture to a dynamic one that accommodates changes in cluster parameters between adjacent time-points. Our models are analagous to existing models such as hierarchical K-means [Steinbach et al., 2000], hierarchical topic models [Blei et al., 2004], dynamic topic models [Blei and Lafferty, 2006a] etc, but are specially suited to model data on unit spheres. This is the first comprehensive work that provides a thorough treatment of vMF distribution in the context of a wide variety of graphical models for data analysis.

1.3 Semi-supervised Structure Expansion

Despite the lack of fully specified structures, end users often have some partial information or preference of topics by which they would like to organize the data. Consider a corpus of news-stories, let us assume a user is interested in organizing the news-stories by regions. Ofcourse the user does not have an exhaustive list of all possible regions in the corpus. However, if the user is willing to identify a few topics in the data for e.g. *U.S., India,* and *China*, it is desirable to have a system that can learn that the user is interested in region-based organization and partition the rest of the unlabeled data by regions and in the process discover other new topics such as *Iraq, Japan, etc*. In other words, we want a system that can generalize the user expectations by modeling shared properties between the observed and unobserved topics (clusters). As another example, consider a collection of speech recordings. In this case, it is perfectly reasonable to organize the recordings either by the content of the recording or by the speaker. An unsupervised clustering algorithm cannot tell which type of partitioning (clustering) is preferable unless the user's expectation is effectively communicated to the system. These problems leads to two challenging questions in the context of clustering:

1. How can we inject supervision to steer the clustering process towards user expectations?
2. If only **some clusters** identified by the user are available as supervision, can the learned user expectations be effectively transferred from the observed clusters to aid in the discovery of **unobserved** (new) clusters in data?

The first problem has been partially addressed by some work in constrained clustering, although not in sufficient depth (see chapter 6). The second problem, to our knowledge, has not been addressed by any work so far. The second problem is particularly important from a practical point of view because realistically users can only label a small set of clusters, whereas the data keeps growing and new clusters are bound to show up. Our solutions to these problem are presented in chapter 6,

- We present a novel probabilistic framework that exploits supervised information in a discriminative and transferable manner to generate better clustering of unlabeled data. The supervision is used to learn a transformation of the data by maximizing the conditional likelihood of the observed cluster labels under the given probabilistic generative clustering process. The estimated transformation function enables us to fold the remaining unlabeled data into a space where new clusters hopefully match user expectations.
- We develop an analagous non-probabilistic framework where the transformation is estimated using a constrained optimization framework instead. The constraints are formulated such that the transformation maximizes the distance between every pair of clusters means by a scale proportional to the sum of the spread of the respective pair of clusters. By learning such a transformation, our hope is that the user expected clusters become well-separated in the transformed space and can be easily recovered by any clustering algorithm.

For both frameworks, we performed extensive testing on more than 20 data sets across several application domains including text, time-series, speech and images and found substantial improvement in performance over existing semisupervised clustering methods as well as unsupervised baselines.

1.4 Key contributions of the thesis

To summarize, the contributions of the thesis are as follows,

1. A Hierarchical Bayesian Logistic Regression framework (HBLR) with associated fast variational inference algorithms for hierarchical classification that significantly improves over state-of-the-art hierarchical and flat baselines (Chapter 2).

2. A large-margin framework that can leverage hierarchical and graph based dependencies using recursive regularization with fast training procedures based on coordinate descent and achieves state-of-the-art results on multiple popular benchmark datasets (Chapter 3).

3. A scalable distributed approach to train Multiclass Logistic Regression model based on upper bounding the log-normalization constant that can scale to tens of thousands of multinomial outcomes (Chapter 4).

4. The first comprehensive framework for structure discovery using the vMF distribution to extract flat, hierarchical and temporal structures in unit normalized data that significantly improves over several popular clustering methods (Chapter 5).

5. Two new frameworks for transformation based clustering methods which can exploit supervision in the form of observed clusters to extract better clusters from unlabeled data and shows significant and consistent improvement over other supervised and unsupervised baselines (Chapter 6).

Finally, in chapter 7, we present some of the interesting directions for future work along these lines.

Bayesian Logistic Models for Hierarchical Classification

2

2.1 Introduction

In this chapter, we develop Hierarchical Bayesian Logistic Regression (HBLR) - a Bayesian framework that can leverage the hierarchical structure between the class-labels for improving classification performance. The Bayesian framework is a natural fit for this problem as it can seamlessly capture the idea that the models at the lower levels of the hierarchy are different forms of specializations of models at ancestor nodes.

We define our model by means of a probabilistic generative process of how the class-labels and the various parameters in the model are generated. More specifically, we define a hierarchical Bayesian model where the prior distribution for the parameters at a node in the hierarchy is a Gaussian centered at the parameters of the parent node. This prior encourages the parameters of nodes that are close in the hierarchy to be similar thereby enabling propagation of information across the hierarchical structure and leading to inductive transfer (sharing statistical strength) among the models corresponding to the different nodes. The strength of the Gaussian prior, and hence the amount of information sharing between nodes, is controlled by its covariance parameter, which is also learned from the data. Modelling the covariance structures gives us the flexibility to incorporate different ways of sharing information in the hierarchy. For example, consider a hierarchical organization of all *animals* with two sub-topics *mammals* and *birds*. By placing feature specific variances, the model can learn that the sub-topic parameters are more similar along common features like 'eyes','claw' and less similar in other sub-topic specific features like 'feathers', 'tail' etc. As another example, the model can incorporate children-specific covariances that allows some sub-topic parameters to be less similar to their parent and some to be more similar; for e.g. sub-topic *whales* is quite distinct from its parent *mammals* compared to its siblings *felines, primates*. Formulating such constraints in

non-Bayesian large-margin approaches is not as easy and to our knowledge has not done before in the context of hierarchical classification.

To make our models be applicable on large-scale structures, we present variational inference methods that are orders of magnitude faster than typical MCMC sampling methods. For high dimensional problems, we develop even faster partial MAP inference methods and associated parallelization schemes that can easily scale to hundreds of thousands of class-labels.

2.2 Related Work

There has been more than a decade of work in hierarchical classification. The most popular in the early stage are the 'pachinko-machine models' [Dumais and Chen, 2000], [Yang et al., 2003] [Liu et al., 2005], [Koller and Sahami, 1997] where the classification task is decomposed into sub-tasks recursively and each node of the hierarchy has an independently trained classifier. The hierarchy is only used to partition the training data and not used any further in the training. The simplicity makes these methods easy to scale, but also makes them limited in effectively using the hierarchical dependencies.

Several approaches have been proposed for making better use of the hierarchical structure. In [Bennett and Nguyen, 2009], a cascading strategy is employed to add the output of lower-level classifiers as additional features for higher-level classifiers. In [DeCoro et al., 2007], a Bayesian aggregation on the results of the individual binary classifiers was proposed. In [Xue et al., 2008], a data-driven pruning strategy is proposed for reducing the size of the original hierarchy. Some improvements over the results of the pachinko-machine models have been reported; however, these approaches are heuristic by nature.

The more principled methods include the large-margin models by [Tsochantaridis et al., 2006], [Cai and Hofmann, 2004],[Rousu et al., 2006],[Dekel et al., 2004], [Cesa-Bianchi et al., 2006] where the discriminant functions take the contributions from all nodes along the path to the root and the model parameters are jointly learned to minimize a global loss over the hierarchy. Similar ideas have been explored in [Zhou et al., 2011] where orthogonality conditions are imposed between the parent and children classifiers and in [McCallum et al., 1998] using Naive Bayes classifiers with hierarchical shrinkage. Although there are a few works that address the scalability of these methods with large number of training instances [Gornitz et al., 2011], [Widmer et al., 2010], there is no work that focuses on scaling with large number of classes. Infact, empirical improvements of most of these methods over simpler approaches have been shown only on small datasets with typically with hundreds (or less) of class-labels. The primary difficulty for *most* of these methods in scaling is due to the high-degree of inter-dependencies among model parameters and the parameters for all the classes cannot be held in memory at

the same time.

Among the Bayesian methods, our approach shares similarity to the correlated Multinomial logit [Shahbaba and Neal, 2007] (corrMNL) in taking Bayesian prior approach to model the hierarchical class structure, but improves over it in two significant aspects - scalability and setting hyperparameters. Firstly, CorrMNL uses slower MCMC sampling for inference, making it difficult to scale to problems with more than a few hundred features and a few hundred nodes in the hierarchy. By modelling the problem as a Hierarchical Bayesian Logistic Regression (HBLR), we are able to vastly improve the scalability by more than 750x (see 2.7.3). Secondly, a difficulty with the Bayesian approaches, that has been largely side-stepped in [Shahbaba and Neal, 2007], is that, when expressed in full generality, they leave many hyperparameters open to subjective input from the user. Typically, these hyper-parameters need to be set carefully as they control the amount of regularization in the model and traditional techniques such as Empirical Bayes or cross-validation encounter difficulties in achieving this. For instance, Empirical Bayes requires the maximization of marginal likelihood which is difficult to compute in hierarchical logistic models [Do et al., 2007] in general, and cross-validation requires reducing the number of free parameters for computational reasons, potentially losing the flexibility to capture the desired phenomena. In contrast, we propose a principled way to set the hyper-parameters directly from data using an approximation to the observed Fisher Information Matrix. Our proposed technique can be easily used to set a large number of hyper-parameters without losing model tractability and flexibility.

2.3 Hierarchical Bayesian Logistic Framework

Define a hierarchy as a set of nodes $\mathcal{N} = \{1, 2...\}$ with the parent relationship $\pi : \mathcal{N} \to \mathcal{N}$ where $\pi(n)$ is the parent of node $n \in \mathcal{N}$. Let $\mathbf{D} = \{(x_i, t_i)\}_{i=1}^{N}$ denote the training data where $x_i \in \mathbb{R}^d$ is an instance, $t_i \in T$ is a label, where $T \subset \mathcal{N}$ is the set of leaf nodes in the hierarchy labeled from 1 to $|T|$. We assume that each instance is assigned to one of the leaf nodes in the hierarchy. If there are any instances assigned to an internal node, spawn a leaf-node under it and re-assign all the instances from the internal node to this new leaf node. Let C_n be the set of all children of node n.

For each node $n \in \mathcal{N}$, we associate a parameter vector w_n which has a Gaussian prior. We set the mean of the prior to the parameter of the parent node, $w_{\pi(n)}$. In what follows, we consider three alternate ways to model the covariance matrix which we call M1, M2 and M3 variants of HBLR. In the M1 variant all the siblings share the same spherical covariance matrix. Formally,

the generative model for M1 is

$$\begin{aligned}
\textbf{M1}\quad & w_{root} \sim \mathcal{N}(w_0, \Sigma_0), \qquad \alpha_{root} \sim \quad \Gamma(a_0, b_0) \\
& w_n |\, w_{\pi(n)}, \Sigma_{\pi(n)} \sim \mathcal{N}(w_{\pi(n)}, \Sigma_{\pi(n)}) \quad \forall n, \qquad \alpha_n \sim \quad \Gamma(a_n, b_n) \quad \forall n \notin T \\
& t \mid x, \mathbf{W} \sim \quad \textit{Categorical}(p_1(x), p_2(x), .., p_{|T|}(x)) \quad \forall (x,t) \in \mathbf{D} \\
& p_i(x) = \quad \exp(w_i^\top x) / \Sigma_{t' \in T} \exp(w_{t'}^\top x)
\end{aligned} \tag{2.1}$$

The parameters of the root node are drawn using user specified parameters w_0, Σ_0, a_0, b_0. Each non-leaf node $n \notin T$ has its own α_n drawn from a Gamma with the shape and inverse-scale parameters specified by a_n and b_n (gamma distribution was chosen as it is conjugate to the precision of the Gaussian). Each w_n is drawn from the Normal with mean $w_{\pi(n)}$ and covariance matrix $\Sigma_{\pi(n)} = \alpha_{\pi(n)}^{-1} I$. The class-labels are drawn from a Multinomial whose parameters are a soft-max transformation of the w_ns from the leaf nodes. This model leverages the class hierarchy information by encouraging the parameters of closely related nodes (parents, children and siblings) to be more similar to each other than those of distant ones in the hierarchy. Moreover, by using different inverse variance parameters α_n for each node, the model has the flexibility to adapt the degree of similarity between the parameters (i.e. parent and children nodes) on a per family basis. For instance it can learn that sibling nodes which are higher in the hierarchy (e.g. *mammals* and *birds*) are generally less similar compared to sibling nodes lower in the hierarchy (e.g. *chimps* and *orangutans*).

Although this model is equivalent to the corrMNL proposed in [Shahbaba and Neal, 2007], the hierarchical logistic regression formulation is different from corrMNL and has a distinct advantage that the parameters can be decoupled. As we shall see in Section 2.4, this enables the use of scalable and parallelizable variational inference algorithms which make our approach 750x faster than [Shahbaba and Neal, 2007].

We can further extend **M1** by allowing the diagonal elements of the covariance matrix $\Sigma_{\pi(n)}$ to be feature-specific instead of uniform. In our previous example with sub-topics *mammals* and *birds*, we may want $w_{mammals}$, w_{birds} to be commonly close to their parent in some dimensions (e.g., in some common features like 'eyes','breathe' and 'claw') but not in other dimensions (e.g., in *bird* specific features like 'feathers' or 'beak'). We accommodate this by replacing prior α_n using $\alpha_n^{(i)}$ for every feature (i). This form of setting the prior is referred to as Automatic Relevance Determination (ARD) and forms the basis of several works such as Sparse Bayesian Learning [Tipping, 2001], Relevance Vector Machines [Bishop and Tipping, 2003], etc. We define the M2 variant of the HBLR approach as:

$$\begin{aligned}
\textbf{M2}\quad & w_n |\, w_{\pi(n)}, \Sigma_{\pi(n)} \sim \mathcal{N}(w_{\pi(n)}, \Sigma_{\pi(n)}) \quad \forall n \\
& \alpha_n^{(i)} \sim \Gamma(a_n^{(i)}, b_n^{(i)}) \quad i = 1..d, \ \forall n \notin T \qquad \text{where } \Sigma_{\pi(n)}^{-1} = \text{diag}(\alpha_{\pi(n)}^{(1)}, \alpha_{\pi(n)}^{(2)}, \ldots, \alpha_{\pi(n)}^{(d)})
\end{aligned}$$

Yet another extension of the M1 model would be to allow each node to have its own covariance matrix for the Gaussian prior over w_n, not shared with its siblings. This enables the model to learn how much the individual children nodes differ from the parent node. For example, consider topic *mammals* and its two sub-topics *whales* and *carnivores*; the sub-topic *whales* is very distinct from a typical *mammal* and is more of an 'outlier' topic. Such mismatches are very typical in hierarchies; especially in cases where there is not enough training data and an entire subtree of topics is collapsed as a single node. M3 aims to cope up with such differences.

$$\begin{aligned} \textbf{M3} \quad w_n|\, w_{\pi(n)}, \Sigma_n &\sim \mathcal{N}(w_{\pi(n)}, \Sigma_n) \qquad \forall n \\ \alpha_n &\sim \Gamma(a_n, b_n) \qquad \forall n \notin T \end{aligned}$$

Note that the only difference between $M3$ and $M1$ is that $M3$ uses $\Sigma_n = \alpha_n^{-1} I$ instead of $\Sigma_{\pi(n)}$ in the prior for w_n. In our experiments we found that M3 consistently outperformed the other variants suggesting that such effects are important to model in hierarchies. Although it would be natural to extend M3 by placing ARD priors instead of the uniform α_n, we do not expect to see better performance due to the difficulty in learning a large number of parameters. Preliminary experiments confirmed our suspicions so we did not explore this direction further.

2.4 Variational Inference

2.4.1 Full Variational Inference

The primary inference procedure in Bayesian methods is to calculate the posterior distribution of the parameters - $\mathbf{W}, \boldsymbol{\alpha}$. We briefly outline the procedure for model M2 and leave the exact details of the inference to Appendix A. For ease of presentation we overload the definition of $\mathbf{D}$ to denote $\mathbf{Y}|\mathbf{X}$ instead of the dataset. The posterior distribution of parameters in model M2, by the Bayes theorem, is given by,

$$p(\mathbf{W}, \boldsymbol{\alpha}|\mathbf{D}) \propto p(\mathbf{D}|\mathbf{W}, \boldsymbol{\alpha}) p(\mathbf{W}, \boldsymbol{\alpha}) \tag{2.2}$$

where

$$p(\mathbf{D}|\mathbf{W}, \boldsymbol{\alpha}) = \prod_{(x,t)\in D} \frac{\exp(w_t^\top x)}{\sum_{t' \in T} \exp(w_{t'}^\top x)} \tag{2.3}$$

$$\begin{aligned} p(\mathbf{W}, \boldsymbol{\alpha}) &= \prod_{n \in \mathcal{N}\setminus T} \prod_{i=1}^{d} p(\alpha_n^{(i)}|a_n^{(i)}, b_n^{(i)}) \prod_{n\in\mathcal{N}} p(w_n|w_{\pi(n)}, \Sigma_{\pi(n)}) \\ &= \prod_{n \in \mathcal{N}\setminus T} \prod_{i=1}^{d} \Gamma(\alpha_n^{(i)}|a_n^{(i)}, b_n^{(i)}) \prod_{n\in\mathcal{N}} \mathcal{N}(w_n|w_{\pi(n)}, \Sigma_{\pi(n)}) \end{aligned} \tag{2.4}$$

The posterior distribution is proportional to the product of a logistic likelihood term (2.3) and the Gamma and Normal prior over $\boldsymbol{\alpha}$, $\boldsymbol{W}$ (2.4). However, this convolution has no closed-form solution, therefore the posterior distribution cannot be computed in closed-form. Therefore we need to develop numerical techniques that can estimate this posterior.

Variational methods are a set of techniques that address this problem by considering a class of *simple distributions* $\mathcal{Q}$ and choosing a member of this class that is closest in KL divergence to the true posterior, that is,

$$\min_{q(\mathbf{W},\boldsymbol{\alpha})\in\mathcal{Q}} \quad KL(\ q(\mathbf{W},\boldsymbol{\alpha})\ ||\ P(\mathbf{W},\boldsymbol{\alpha}|\mathbf{D})\ ) \tag{2.5}$$

Expanding the KL divergence,

$$\begin{aligned} KL(\ q(\mathbf{W},\boldsymbol{\alpha})\ ||\ P(\mathbf{W},\boldsymbol{\alpha}|\mathbf{D})\ ) &= \int q(\mathbf{W},\boldsymbol{\alpha}) \log \frac{q(\mathbf{W},\boldsymbol{\alpha})}{P(\mathbf{W},\boldsymbol{\alpha}|\mathbf{D})} dq \\ &= \int q(\mathbf{W},\boldsymbol{\alpha}) \log q(\mathbf{W},\boldsymbol{\alpha}) - \int q(\mathbf{W},\boldsymbol{\alpha}) \log \frac{P(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})}{P(\mathbf{D})} \\ &= P(\mathbf{D}) - (E_q\left[\log P(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] - E_q\left[\log q(\mathbf{W},\boldsymbol{\alpha})\right]) \end{aligned}$$

Note that $P(\mathbf{D})$ does not depend on q, therefore minimizing the KL divergence in (2.5) can be reformulated as the following equivalent maximization problem,

$$\max_{q(\mathbf{W},\boldsymbol{\alpha})\in\mathcal{Q}} \quad E_q\left[\log P(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] - E_q\left[\log q(\mathbf{W},\boldsymbol{\alpha})\right] \tag{2.6}$$

The objective in (2.6) is also known as the evidence lower bound (ELBO) since it lower bounds the likelihood of the data (see [Wainwright and Jordan, 2008] for more details). To make this optimization tractable, the class of *simple distributions* $\mathcal{Q}$ that we consider are a product of independent normal and gamma distributions. More specifically,

$$\begin{aligned} q(\mathbf{W},\boldsymbol{\alpha}) &= \prod_{n\in\mathcal{N}\setminus T} q(\alpha_n) \prod_{n\in\mathcal{N}} q(w_n) \\ q(w_n) &\propto \mathcal{N}(.|\mu_n,\Psi_n) \\ q(\alpha_n) &= \prod_{i=1}^{d} q(\alpha_n^{(i)}) \propto \prod_{i=1}^{d} \Gamma(.|\tau_n^{(i)},\upsilon_n^{(i)}) \end{aligned}$$

Here $\boldsymbol{\tau},\boldsymbol{\upsilon},\boldsymbol{\mu},\boldsymbol{\Psi}$ are the parameters of this posterior that we need to estimate such that the ELBO is maximized. To perform the maximization, we will use the coordinate ascent technique where we update each parameter of the posterior to optimize the ELBO keeping the rest of the parameters fixed. The update rules for the parameters are given by (the exact details of the

derivation are presented in the appendix A),

$$\Psi_n^{-1} = I(n \in T) \sum_{(x,t)\in D} 2\lambda(\xi_{xn}) x x^\top + \text{diag}(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}) + |C_n| \,\text{diag}(\frac{\tau_n}{\upsilon_n}) \tag{2.7}$$

$$\mu_n = \Psi_n \left(I(n \in T) \sum_{(x,t)\in D} (I(t = n) - \frac{1}{2} + 2\lambda(\xi_{xn})\beta_x)x + \text{diag}(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}})\mu_{\pi(n)} + \text{diag}(\frac{\tau_n}{\upsilon_n}) \sum_{c\in C_n} \mu_c \right) \tag{2.8}$$

$$\upsilon_n^{(i)} = b_n^{(i)} + \sum_{c\in C_n} \Psi_n^{(i,i)} + \Psi_c^{(i,i)} + (\mu_n^{(i)} - \mu_c^{(i)})^2 \tag{2.9}$$

$$\tau_n^{(i)} = a_n^{(i)} + \frac{|C_n|}{2} \tag{2.10}$$

2.4.2 Partial MAP Inference

In cases where the number of dimensions is more than a few thousands, the requirement for a matrix inversion of Ψ_n in step (2.7) to (2.8) could be demanding. In such scenarios, we split the inference into two stages, first calculating the posterior of w_n (for the leaf nodes) using MAP solution, and second calculating the posterior of α_n. In the first stage, we find the MAP estimate w_n^{map} (for $n \in T$ or $n \in \mathcal{N}$) and then use laplace approximation to approximate the posterior using a separate Normal distribution for each dimension, thereby leading to a diagonal covariance matrix. Note that due to the laplace approximation, w_n^{map} and the posterior mean μ_n coincide.

$$w^{map} = \arg\max_{\mathbf{W}} \sum_{n\in T} -\frac{1}{2}(w_n - w_{\pi(n)})^\top \,\text{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right)(w_n - w_{\pi(n)}) + \log p(\mathbf{D}|\mathbf{W}, \boldsymbol{\alpha}) \tag{2.11}$$

$$(\Psi_n^{(i,i)})^{-1} = \sum_{(x,t)\in D} x^{(i)} p_{xn}(1 - p_{xn}) x^{(i)} + \text{diag}\left(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}\right) \tag{2.12}$$

where p_{xn} is the probability that training instance x is labeled as n (using the soft max probability given in (2.3)). The $\arg\max$ in (2.11) can be computed for all w_n at the same time using optimization techniques like LBFGS [Liu and Nocedal, 1989]. For the second stage, parameters τ_n and υ_n are updated using (2.9), (2.10). Full MAP inference is also possible by performing an alternating maximization between w_n, α_n but we do not recommend it as there is no gain in scalability compared to partial MAP Inference and it loses the posterior distribution of α_n.

2.4.3 Parallel Inference

For large hierarchies, it might be impractical to learn the parameters of all classes, or even store them in memory on a single machine. We therefore, devise a parallel memory-efficient implementation scheme for our partial MAP Inference. There are 4 sets of parameters that are updated - $\{\mu_n, \Psi_n, \tau_n, \nu_n\}$. The Ψ_n, τ_n, ν_n can be updated in parallel for each node using (2.12), (2.9), (2.10).

For μ, the optimization step in (2.11) is not easy to parallelize since the w's are coupled together inside the soft-max function. To make it parallelizable we replace the soft-max function in (2.1) with multiple binary logistic functions (one for each terminal node), which removes the coupling of parameters inside the log-normalization constant. The optimization can now be done in parallel by making the following observations - firstly note that the optimization problem in (2.11) is concave maximation, therefore any order of updating the variables reaches the same unique maximum. Secondly, note that the interactions between the w_n's are only through the parent and child nodes. By fixing the parameters of the parent and children, the parameter w_n of a node can be optimized independently of the rest of the hierarchy. One simple way to parallelize is to traverse the hierarchy level by level, optimize the parameters at each level in parallel and iterate until convergence. A better way that achieves a larger degree of parallelization is to iteratively optimize the odd and even levels - if we fix the parameters at the odd levels, the parameters of parents and the children of all nodes at even levels are fixed and the w_n's at all even levels can be optimized in parallel. The same goes for optimizing the odd level parameters. To aid convergence we interleave the μ, Ψ updates with the τ, ν updates and warm-start with the previous value of μ_n. In practice, for the larger hierarchies we observed speedups linear in the number of processors. Note that the convergence follows from viewing this procedure as block coordinate ascent on a concave differentiable function [Luo and Tseng, 1992].

2.5 Estimating Data-dependent Priors

The w_0, Σ_0 represent the overall mean and covariance structure for the w_n. We set $w_0 = 0$ and $\Sigma_0 = I$ because of their minimal effect on the rest of the parameters. The $a_n^{(i)}, b_n^{(i)}$ are variance components such that $\frac{b_n^{(i)}}{a_n^{(i)}}$ represents the expected variance of the $w_n^{(i)}$. Typically, choosing these parameters is difficult before seeing the data. The traditional way to overcome this is to learn $\{a_n, b_n\}$ from the data using Empirical Bayes. Unfortunately, in our proposed model, one cannot do this as each $\{a_n, b_n\}$ is associated with a single α_n. Generally, we need more than one sample value to learn the prior parameters effectively [Casella].

We therefore resort to a data dependent way of setting these parameters by using an approximation to the observed Fisher Information matrix. We first derive on a simpler model and then extend it to a hierarchy. Consider the following binary logistic model with unknown w and let the Fisher Information matrix be I and observed Fisher Information $\hat{I}$. Given some training instances D_{binary}

$$\begin{aligned} Y \mid x &\sim Bernoulli\left(\frac{\exp(w^\top x)}{1+\exp(w^\top x)}\right) \\ I &= E\left[p(x)(1-p(x))xx^\top\right] \\ \hat{I} &= \sum_{(x,t)\in D_{binary}} \hat{p}(x)(1-\hat{p}(x))xx^\top \end{aligned} \tag{2.13}$$

It is well known that I^{-1} is the asymptotic covariance of the MLE estimator of w, so a reasonable guess for the covariance of a Gaussian prior over w could be the observed $\hat{I}^{-1}$ from a dataset D_{binary}. The problem with $\hat{I}^{-1}$ is that we do not have a good estimate $\hat{p}(x)$ for a given x as we have exactly one sample for a given x i.e each instance x is labeled exactly once with certainty, therefore $\hat{p}(x)(1-\hat{p}(x))$ will always be zero. Therefore we approximate $\hat{p}(x)$ as the sample prior probability independent of x, i.e. $\hat{p}(x) = \hat{p} = \Sigma_{(x,t)\in D}\frac{t}{|D|}$. Now, the prior on the covariance of w can be set such that the expected covariance is $\hat{I}^{-1}$.

To extend this to hierarchies, we need to handle multiple classes, which can be done by estimating $\hat{I}(n)^{-1}$ for each $n \in T$, as well handle multiple levels, which can be done by recursively setting a_n, b_n as follows,

$$(a_n^{(i)}, b_n^{(i)}) = \begin{cases} (\sum_{c\in C_n} a_c^{(i)}, \sum_{c\in C_n} b_c^{(i)}) & \text{if } n \notin T \\ (1, \hat{I}(n)^{-1(i,i)}) & \text{if } n \in T \end{cases}$$

where $\hat{I}(n)$ is the observed Fisher Information matrix for class label n. This way of setting the priors is similar to the method proposed in [Kass and Natarajan, 2006], the key differences are in approximating $p(x)(1-p(x))$ from the data rather using $p(x) = \frac{1}{2}$, extension to handle multiple classes as well as hierarchies.

We also tried other popular strategies such as setting improper gamma priors $\Gamma(\epsilon, \epsilon)$ $\epsilon \to 0$ widely used in many ARD works (which is equivalent to using type-2 ML for the α's if one uses variational methods [Bishop et al., 2006]) and Empirical Bayes using a single a and b (as well as other Empirical Bayes variants). Neither of worked well, the former being to be too sensitive to the value of ϵ which is in agreement with the observations made by [Gelman, 2006] and the latter constraining the model by using a single a and b.

Table 2.1: Dataset Statistics

Dataset	#Training	#Testing	#Class-Labels	#Leaf-labels	Depth	#Features
CLEF	10,000	1,006	87	63	4	89
NEWS20	11260	7505	27	20	3	53975
IPC	46,324	28,926	552	451	4	541,869
LSHTC-small	4,463	1,858	1,563	1,139	6	51,033
DMOZ-2010	128,710	34,880	15,358	12,294	6	381,580

2.6 Experimental Settings

2.6.1 Datasets

We used the following four datasets for evaluating the performance of our proposed HBLR models,

1. **CLEF** [Dimitrovski et al., 2011] A hierarchical collection of medical X-ray images with image descriptor features. The dataset was normalized to have zero mean and unit variance to improve convergence.

2. **NEWS20** [1] A collection of newsgroup posts by users across 20 newsgroup categories.

3. **IPC** [WIPO] A collection of patents organized according to the International Patent Classification Hierarchy. We considered prediction at the subclass level of the hierarchy.

4. **LSHTC-small**, **DMOZ-2010** The small and the larger web-page collections released as a part of the LSHTC (Large-Scale Hierarchical Text Classification) evaluation 2010 [2]. It is essentially a subset of the web pages from the Open Directory Project [3].

For all the text datasets, we did the standard preprocessing steps of removing stop words, stemming followed by Tf-Idf (*ltc*) term weighting [Manning et al., 2008]. Note that all the instances in the dataset have exactly one ground-truth class-label which belongs to one of the leaf-nodes of the hierarchy.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/
[2] http://lshtc.iit.demokritos.gr/node/3
[3] http://dmoz.org

2.6.2 Description of Baselines

We compared our HBLR models against the following state-of-the-art hierarchical and non-hierarchical models,

1. **Flat Baselines**: These baselines do not make use of the hierarchy to estimate the parameters of the classifiers,

   - *One-versus-rest methods*: We compared against the most popular Binary Support Vector Machines (**BSVM**) and Binary Logistic Regression (**BLR**). The classifiers are learnt independently for each class.
   - *Multiclass classifiers*: We used the multiclass versions of support vector machines (**MSVM**) and logistic regression (**MLR**). The classifiers are learnt such that the prediction score of the ground-truth class-label is higher than every other incorrect class-label.

2. **Hierarchical Baselines**: These baselines use the hierarchical information to encode dependencies into the model parameters.

   - **Hierarchical SVM (HSVM)** [Tsochantaridis et al., 2006] This method is one of the most popular large-margin discriminative method for structured output prediction. We use the specific instatiation of this framework developed for hierarchical classification. The model defines a hierarchical path dependent discriminant function that minimizes a global hierarchical loss. More specifically, the parameters of the model are estimated by maximizing the margin between every pair of correct and incorrect labels with a penalty proportional to the hierarchical distance between the labels. The resulting optimization problem is solved by using a constraint generation framework that repeatedly adds the most violated constraint to the working set.
   - **Hierarchical Orthogonal Transfer (OT)** [Zhou et al., 2011]: OT is large-margin method that encourages the classifiers at each node of the hierarchy to be different from the classifiers at its ancestors. Specifically, the regularization term is modified to enforce orthogonality between the parameters of parent and the children. The resulting optimization problem is solved using regularized dual averaging method.
   - **Top-down SVM (TD)** [Liu et al., 2005], [Dumais and Chen, 2000], [Koller and Sahami, 1997] We employed the simplest variant of pachinko machine style top-down method using support vector machine classifier. This is a popular baseline in several previous works.

3. **Bayesian Baseline**:

- **Correlated Multinomial Logit (CorrMNL)** [Shahbaba and Neal, 2007]: [4]: This method uses a Bayesian form of the multinomial logit model with a prior that introduces correlations between the parameters for classes that are nearby in the hierarchy. However, the model suffers from two important limitations; Firstly, sensitive hyperparameters of the model need to be tweaked and manually set by the user. This is hard, especially if the user has no prior information. Secondly, the inference is performed using MCMC sampling and therefore cannot scale to large number of classes. Due to this limitation, we report the results of this method only on the smallest CLEF dataset.

For all the non-Bayesian methods, we tune the regularization parameter using 5 fold cross-validation with a range of values from 10^{-5} to 10^{5}. For the HBLR models, denoted as {M1,M2,M3}-map, we used partial MAP Inference for all the datasets (except CLEF) because full variational inference is not scalable to high dimensions. The IPC and DMOZ-2010 are very large datasets so we are unable to test any method other than our parallel implementation of HBLR, and TD, BLR, BSVM which can be trivially parallelized.

2.6.3 Evaluation Metrics

For all the datasets, we used the standard Micro-F_1 and Macro-F_1 evaluation metric (see Appendix B for definitions).

2.7 Results

We present and discuss a partial set of results that establish the better performance of our proposed HBLR models against several hierarachical and flat baselines. Section 3.5 contains several additional results on benchmark datasets as well as extension of HBLR to multilabel datasets.

2.7.1 Comparison with Flat Baselines

Table 2.2 reports the results of our proposed models against the flat baselines - BSVM, MSVM, BLR and MLR. The results show that on all the datasets except NEWS20 and the Micro-F_1 on DMOZ-2010, our proposed models perform better than the flat baselines.

To validate the results further, we conducted significance tests using sign-test for Micro-F_1 and a wilcoxon rank test on the Macro-F_1 scores. For every data collection, each method is

[4] http://www.ics.uci.edu/ babaks/Site/Codes.html

Table 2.2: **Comparison against flat baselines**: Macro-F_1 and Micro-F_1 on 5 datasets. Bold faced number indicates best performing method. NS denotes method is not scalable to the dataset. The significance-test results between the best performing best on each dataset against the rest of the methods are denoted by a $*$ for a p-value less than 5% and † for p-value less than 1%.

	Flat Methods				HBLR models		
	BSVM	**BLR**	**MSVM**	**MLR**	**M1-map**	**M2-map**	**M3-map**
CLEF							
Macro-F_1	48.59†	53.26†	54.33†	54.76†	55.53†	54.76†	**59.65**
Micro-F_1	77.53†	79.92†	80.02†	80.52†	80.88*	80.25*	**81.41**
NEWS20							
Macro-F_1	**82.32**	82.17	81.73	81.82	81.54	80.91*	81.69
Micro-F_1	**83.10**	82.97	82.47*	82.56*	82.24*	81.54*	82.56*
IPC							
Macro-F_1	45.71†	48.29†	NS	NS	50.43†	47.45†	**51.06**
Micro-F_1	53.12†	55.03†			55.80*	54.22†	**56.02**
LSHTC-small							
Macro-F_1	28.62*	28.12†	28.34*	28.38*	28.81*	25.81†	**30.81**
Micro-F_1	45.21*	44.94†	45.62	45.20	45.48	43.31†	**46.03**
DMOZ-2010							
Macro-F_1	**32.64**	31.58	NS	NS	30.29	29.45	31.88
Micro-F_1	45.36	45.40			45.10	42.04	**45.64**

compared to the best performing method on that dataset. The null hypothesis is that there is no significant difference between the two systems being compared, the alternative is that the best performing method is better. The results of the significance test show that on most datasets the results are statistically significant. Note that we are unable to conduct significance tests on DMOZ-2010 since we did not have access to class-wise performance scores and true-test labels - our reported evaluation measures on these datasets are from the output of an online evaluation system which does not reveal classwise performance measures nor true test labels.

Among the models M1,M2 and M3, the performance of M3 seems to be consistently better than M1, followed by M2. Although M2 is more expressive than M1, the benefit of a better model seems to be offset by the difficulty in learning a large number of parameters.

Table 2.3: **Comparison against hierarchical baselines**: Macro-F_1 and Micro-F_1 on 5 datasets. Bold faced number indicates best performing method. NS denotes method is not scalable to the dataset. The significance-test results between the best performing method on each dataset against the rest of the methods are denoted by a $*$ for a p-value less than 5% and † for p-value less than 1%.

	Hierarchical Methods			HBLR models		
	HSVM	**OT**	**TD**	**M1-map**	**M2-map**	**M3-map**
CLEF						
Macro-F_1	57.23*	37.12†	32.32†	55.53†	54.76†	**59.65**
Micro-F_1	79.92†	73.84†	70.11†	80.88*	80.25*	**81.41**
NEWS20						
Macro-F_1	80.04†	81.20	81.86	81.54	80.91	**81.69**
Micro-F_1	80.79*	81.98*	81.20†	82.24	81.54	**82.56**
IPC						
Macro-F_1	NS	NS	42.62†	50.43†	47.45†	**51.06**
Micro-F_1	NS	NS	55.34†	55.80*	54.22†	**56.02**
LSHTC-small						
Macro-F_1	21.95†	19.45†	20.01†	28.81*	25.81†	**30.81**
Micro-F_1	39.66†	37.12†	38.48†	45.48	43.31†	**46.03**
DMOZ-2010						
Macro-F_1	NS	NS	22.30†	30.29L	29.45	**31.88**
Micro-F_1	NS	NS	38.46†	45.10	42.04	**45.64**

2.7.2 Comparison with Hierarchical Baselines

Comparing to the other hierarchical baselines (Table 2.3), M3 achieves significantly higher performance on all datasets, showing that the Bayesian approach is able to leverage the information provided in the class hierarchy. Among the baselines, we find that the average performance of HSVM is higher than the TD, OT. This can be partially explained by noting that both OT and TD are greedy top-down classification methods and any error made in the top level classifications propagates down to the leaf node; in contrast to HSVM which uses an exhaustive search over all labels. However, the result of OT do not seem to support the conclusions in [Zhou et al., 2011]. We hypothesize two reasons - firstly, the orthogonality condition which is assumed in OT does not hold in general, secondly, unlike [Zhou et al., 2011] we use cross-validation to set the underlying regularization parameters rather than setting them arbitrarily to 1 (which was used in [Zhou et al., 2011]).

Surprisingly, the hierarchical baselines (HSVM,TD and OT) experience a very large drop in

Table 2.4: **Comparison with CorrMNL:** Macro-F_1 and Micro-F_1 of the HBLR models and CorrMNL the **CLEF** dataset

		{M1,M2,M3}-var			{M1,M2,M3}-map			{M1,M2,M3}-flat		
	CorrMNL	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
Macro-f1	55.59	56.67	51.23	**59.67**	55.53	54.76	59.65	52.13	48.78	55.23
Micro-f1	81.10	81.21	79.92	**81.61**	80.88	80.25	81.41	79.82	77.83	80.52
Time (mins)	2279	79	81	80	3	3	3	3	3	3

performance on LSHTC-small when compared to the flat baselines, indicating that the hierarchy information actually mislead these methods rather than helping them. In contrast, M3 is consistently better than the flat baselines on all datasets except NEWS20. In particular, M3 performs significantly better on the largest datasets, especially in Macro-F_1, showing that even very large class hierarchies can convey very useful information, highlighting the importance of having a scalable, parallelizable hierarchical classification algorithm.

2.7.3 Comparison of Bayesian methods

First, to evaluate the speed advantage of the variational inference, we compare the full variational {M1,M2,M3}-var and partial MAP {M1,M2,M3-map} inference [5] for the three variants of HBLR to the MCMC sampling based inference of CorrMNL [Shahbaba and Neal, 2007]. For CorrMNL, we used the implementation as provided by the authors[6]. We performed sampling for 2500 iterations with 1000 for burn-in. Re-starts with different initialization values gave the same results for both MCMC and variational methods. All models were run on a single CPU without parallelization. We used the small CLEF [Dimitrovski et al., 2011] dataset in order to be able to run CorrMNL model in reasonable time. The results are presented in Table 2.4. For an informative comparison, we also included the results of {M1,M2,M3}-flat, our proposed approach using a flat hierarchy (i.e. no hierarchical information).

With regards to scalability, partial MAP inference is the most scalable method being orders of magnitude faster (750x) than CorrMNL. Full variational inference, although less scalable as it requires $O(d^3)$ matrix inversions in the feature space, is still orders of magnitude faster (20x) than CorrMNL. In terms of performance, we see that the partial MAP inference for the HBLR has only small loss in performance compared to the full variational inference while having similar training time to the flat approach that does not model the hierarchy ({M1,M2,M3}-flat).

[5] Code available at http://gcdart.blogspot.com/
[6] http://www.ics.uci.edu/~babaks/Site/Codes.html

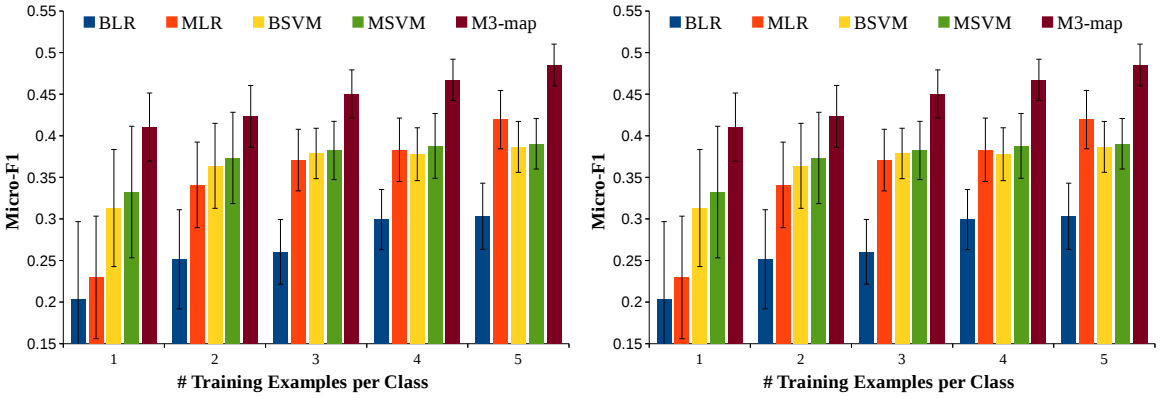


Figure 2.1: Micro-F_1 (left) & Macro-F_1 (right) on the CLEF dataset with limited number of training instances.

2.7.4 Effect of Training-set Sizes

To further establish the importance of modeling the hierarchy, we test our approach under scenarios when the number of training instances is limited. We expect the hierarchy to be most useful in such cases as it enables of sharing of information between class parameters. To verify this, we progressively increased the number of training instances per class-label on the CLEF dataset and compared M3-map with the other best performing methods. Figure 2.1 reports the results of M3-map, MLR, BSVM, MSVM averaged over 20 runs. The results shows that M3-map is significantly better than the other methods especially when the number of instances is small. For instance, when there is exactly one training example per class, M3-map achieves a whopping 10% higher Micro-F_1 and a 2% higher Macro-F_1 than the next best method. We repeated the same experiments on the NEWS20 dataset but however did not find an improved performance even with limited training instances suggesting that the hierarchical methods are not able to leverage the hierarchical structure of NEWS20.

2.8 Summary

In this chapter we presented a HBLR approach to hierarchical classification. We formulated three ways of unraveling dependencies between class-labels by modelling the covariance structure between the parameters in different ways. The model M3, which learns a node-specific covariance seems to show the best performance among the three models.

For all the models we developed fast variational methods, partial MAP inference methods and associated parallelization schemes to tackle classification with tens of thousands (and more)

of class-labels. We developed and tested a practical way to set hyperparameters instead of using vague priors or putting the onus on the end user to set parameters. Our experiments on five datasets (and more in the next chapter) show that HBLR significantly outperforms other popular hierarchical classification methods.

Recursive Regularization for Graphical and Hierarchical Classification

3

3.1 Introduction

In general, not all class-label dependencies are hierarchical (for e.g. the dependencies among Wikipedia categories are in the given in the form of a graph). To our knowledge, there has been no work that addresses the problem of classification with graphical dependencies between class-labels. The Hierarchical Bayesian approach developed in the previous chapter cannot be generalized to graphs as there is no notion of parent or child.

In this chapter, we develop the *recursive regularization* framework for large-scale classification that can leverage both hierarchical and graphical dependencies between the class-labels for improving classification. The framework is generally applicable to both large-margin classifiers (like support vector machines) as well as probabilistic classifiers (like logistic regression). The dependencies among classes and sub-classes are used to define a joint objective for regularization of model parameters; in the case of a hierarchy, the model parameters of the siblings nodes who share the same parent are regularized towards the common parent node and in the case of a graph, the model parameter of a node are regularized towards each of its neighbors. Intuitively, it is based on the assumption that the nearby classes in the hierarchy or graph are semantically close to each other and hence share similar model parameters. Note that this model is simpler than the fully Bayesian models such correlated Multinomial Logit [Shahbaba and Neal, 2007] and the HBLR models (chapter 2) where the dependencies are modeled in richer forms, controlling both the means and covariances in Gaussain models. However it is the simplicity of the approach makes it easier to scale than the fully Bayesian approaches.

For the scalability of our method, we develop a parallel and iterative coordinate descent scheme that can easily tackle the large datasets with millions of dimensions and hundreds

of thousands of classes. The key idea here is to formulate the coordinate descent objective function in a way that the dependencies between the various parameters can be easily localized and the computations for all the local regions can be carried out in parallel. Moreover, the local computations at leaf nodes are dualized in order side-step non-differentiability issues when using loss functions such as Hinge loss. As we shall see in section 3.5, this combination of using iterative parallelization of local computations and fast dual coordinate descent methods for each local computation leads to optimization schemes that can easily scale to very large datasets.

We tested our proposed methods in terms effectiveness as well efficiency by conducting evaluations against other state-of-the-art methods on nine benchmark datasets including the large-scale datasets from the Large-scale Hierarchical Text Classification Challenge [1]. With respect to effectiveness, we found that our methods outperformed all the other methods on most tested datasets. With respect to efficiency, we show for the first time that global optimization with hierarchies and graphs can be efficiently computed for the largest datasets such as wikipedia with 300,000 classes and 2 Million training instances in a matter of 37 hours. Experiments showed state-of-the-art results on multiple benchmark datasets which are orders of magnitudes ($\sim$ 1000x) larger than what the most existing methods have been scaled to [Tsochantaridis et al., 2006],[Cai and Hofmann, 2004], [Rousu et al., 2006], [Dekel et al., 2004], [Cesa-Bianchi et al., 2006], [Widmer et al., 2010].

Indirectly related to our paper are a few works in multitask learning [Evgeniou and Pontil, 2004], [Argyriou et al., 2008],[Argyriou et al., 2007], [Widmer et al., 2010] where regularization was used as a tool to share information between tasks. However, their focus is not scalability and their techniques cannot be directly applied to to problems with hundreds of thousands of class-labels. Other works include regularization on graphs such as [Zhang et al., 2006], [Smola and Kondor, 2003], but the focus is on graphical dependencies between the instances and not between class-labels.

3.2 Recursive Regularization Framework

The key idea in recursive regularization is to incorporate the class-label dependencies into the regularization structure of the parameters. To formulate the regularization framework, we resort to the Structural Risk Minimization framework which prescribes choosing a prediction function to minimize a combination of the empirical risk on the training dataset and a regularization term to penalize the complexity of the function. We first describe the framework for hierarchies and then extend it to graphs. We reuse the notation from chapter 2 where a hierar-

[1] http://lshtc.iit.demokritos.gr/

chy is defined over a set of nodes $\mathcal{N} = \{1, 2...\}$ using a parent relationship $\pi : \mathcal{N} \rightarrow \mathcal{N}$ where $\pi(n)$ is the parent of node $n \in \mathcal{N}$ and let $T \subset \mathcal{N}$ denote the set of leaf nodes in the hierarchy labeled from 1 to $|T|$. Let $x_i \in \mathbb{R}^d$ denote the i'th training instance and $y_{in} \in \{+1, -1\}$ denote the label of x_i w.r.t node n, i.e. whether x_i belongs to node n or not. Note that unlike HBLR, each instance can belong to one or more leaf nodes in the hierarchy.

The prediction function in the context of hierarchies is parameterized by $\mathbf{W} = \{w_n : n \in \mathcal{N}\}$ and the empirical risk is defined to be the loss incurred by the instances at the leaf-nodes of the hierarchy using a loss function L. The parameters are estimated by minimizing,

$$\arg\min_{\mathbf{W}} \lambda(\mathbf{W}) + C \sum_{n \in T} \sum_{i=1}^{N} L(y_{in}, x_i, w_n) \tag{3.1}$$

where $\lambda(\mathbf{W})$ denotes the regularization term and C is a parameter that controls how much to fit to the training data. We propose to use the hierarchy in the learning process by incorporating a recursive structure into the regularization term for $\mathbf{W}$. More specifically, the regularization term is

$$\lambda(\mathbf{W}) = \arg\min_{\mathbf{W}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||w_n - w_{\pi(n)}||^2$$

This recursive form of regularization enforces the parameters of the node to be similar to the parameters of its parent under euclidean norm. Intuitively, it models the hierarchical dependencies in the sense that it encourages parameters which are nearby in the hierarchy to be similar to each other. This helps classes to leverage information from nearby classes while estimating model parameters and helps share statistical strength across the hierarchy. In the case of graphs, the parameters of each node in the graph is regularized towards each of its neighbours (instead of its parent and children). Specifically, given a graph with a set of edges $E \subseteq \{(i, j) : i, j \in \mathcal{N}\}$, the regularization term is given by

$$\lambda(\mathbf{W}) = \sum_{(i,j) \in E} \frac{1}{2} ||w_i - w_j||^2$$

By defining L to be the hinge-loss, we get a large-margin framework (RR-SVM) [2] and by defining L to be the logistic loss, we get the probabilistic framework (RR-LR). The key advantage of RR-{SVM,LR} over other hierarchical models such as [Tsochantaridis et al., 2006], [Cai and Hofmann, 2004], [Zhou et al., 2011] is that there are no constraints that maximizes the margin

[2] In our previous work [Gopal and Yang, 2013b],[Gopal et al., 2012], the methods were denoted by HR-{SVM,LR}. Here we will use the notation RR-{SVM,LR} for convenience.

between correct and incorrect predictions. This keeps the dependencies between the parameters minimal and in turn enables us to develop a parallel-iterative method to optimize the objective thereby scaling to very large problems.

3.3 Large-scale Training of RR Models

3.3.1 RR-SVM

The primary optimization problem in RR-SVM is to estimate the parameter **W**,

$$\min_{\mathbf{W}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||w_n - w_{\pi(n)}||^2 + C \sum_{n \in \mathcal{T}} \sum_{i=1}^{N} (1 - y_{in} w_n^\top x_i)_+ \tag{3.2}$$

We resort to an iterative approach where we update the parameters associated with each node n iteratively by fixing the rest of the parameters. To tackle the non-differentiability in some of the updates (i.e. the updates at the leaf-nodes), we convert these sub-problems into their dual form which is differentiable and optimize it using coordinate descent. For each non-leaf node $n \notin T$, differentiating the objective of eq (3.2) w.r.t w_n yields a closed-form update for w_n given by,

$$w_n = \frac{1}{|C_n| + 1} \left(w_{\pi(n)} + \sum_{c \in C_n} w_c \right) \tag{3.3}$$

For each leaf node $n \in T$, the objective cannot be differentiated due to a discontinuous hinge loss function. By isolating the terms that depend on w_n and introducing slack variables ξ_{in}, the primal objective of the subproblem for w_n is given by,

$$\begin{aligned} \min_{w_n} \quad & \frac{1}{2} ||w_n - w_{\pi(n)}||^2 + C \sum_{i=1}^{N} \xi_{in} \\ \text{subject to} \quad & \xi_{in} \geq 0 && \forall i = 1..N \\ & \xi_{in} \geq 1 - y_{in} w_n^\top x_i && \forall i = 1..N \end{aligned}$$

The dual problem of the above primal subproblem (by introducing appropriate dual variables α_i , $i = 1..N$) is

Algorithm 1 Optimization of RR-SVM and RR-LR

Input : **D**, C, π,T, $\mathcal{N}$
Result : weight vectors $\mathbf{W}^*$

While Not Converged
  For each $n \in \mathcal{N}$
    If $n \notin T$
      Update w_n of the non-leaf node using equation (3.3)
    Else
      If solving **RR-SVM**
        1. Solve the dual optimization problem (3.4)
        2. Update the primal parameters w_n using equation (3.5)
      Else If solving **RR-LR**
        1. Solve optimization problem (3.8) using LBFGS
      End
    End
  End

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_{in}y_{jn}x_i^\top x_j - \sum_{i=1}^{N}\alpha_i(1 - y_{in}w_{\pi(n)}^\top x_i) \tag{3.4}$$
$$0 \leq \alpha_i \leq C$$

To solve this subproblem, one can try to use second order methods such as interior-point methods etc. The downside of such solvers is that it takes a long time even for a single iteration and requires the entire kernel matrix of size $O(N^2)$ to be stored in memory. Typical large-scale problems have at least hundreds of thousands of instances and the memory required to store the kernel matrix is in the order of hundreds of gigabytes for each class thereby rendering it impractical. Instead we propose a coordinate descent approach which has minimal memory requirements and converges quickly even for large problems. Our work is based on the dual coordinate descent developed in [Hsieh et al., 2008].

The core idea in coordinate descent is to iteratively update each dual variable α_i. In the objective function eq (3.4), the update for each dual variable has a simple closed form solution. To derive the update for the i^{th} dual variable α_i given by $\alpha_i + d$, we solve the following one-variable problem,

$$\begin{aligned} \min_{d} \quad & \frac{1}{2} d^2 (x_i^\top x_i) + d \left(\sum_{i=1}^{N} \alpha_i y_{in} x_i \right)^\top x_i - d(1 - y_{in} w_{\pi(n)}^\top x_i) \\ \text{subject to} \quad & 0 \leq \alpha_i + d \leq C \end{aligned}$$

Basically, we substituted α_i by $\alpha_i + d$ in (3.4) and discarded all the terms that do not depend on d. This one-variable problem can be solved in closed form by considering the gradient of the objective and appropriately updating α_i to obey the constraints. The gradient G for the above objective and the corresponding update for α_i is given by,

$$\begin{aligned} G &= a'^\top x_i - 1 + y_{in} w_{\pi(n)}^\top x_i \\ \alpha_i^{new} &= \min \left(\max \left(\alpha_i^{old} - \frac{G}{x_i^\top x_i}, 0 \right), C \right) \end{aligned}$$

where $a' = \sum_{i=1}^{N} \alpha_i y_{in} x_i$ is an auxilary variable maintained and updated throughout the optimization of the subproblem. The time complexity for each α_i update is *O(#nnz in x_i)* - the number of non-zero dimensions in x_i and the memory requirement for solving the entire subproblem is $O(N)$ - far more efficient than that $O(N^2)$ compared to the second order methods. From the solution of the dual subproblem, the update for w_n in the primal form can be derived from the K.K.T conditions for the subproblem,

$$w_n = w_{\pi(n)} + \sum_{i=1}^{N} \alpha_i y_{in} x_i \tag{3.5}$$

Note that the convergence of the above optimization method can be derived by viewing the procedure as a block coordinate descent scheme on a convex function where the blocks corresponds to parameters at each node of the hierarchy [Luo and Tseng, 1992], [Tseng, 2001].

For graphs, we employ a similar block coordinate descent algorithm by iteratively optimizing each parameter w_n. The optimization subproblem at node n is given by,

$$\min_{w_n} \quad \frac{1}{2} \sum_{j:(n,j)\in E} ||w_n - w_j||^2 + C \sum_{i=1}^{N} (1 - y_{in} w_n^\top x_i)_+ \tag{3.6}$$

In order to derive the dual of the above subproblem, we first re-write it by expanding the regularization term and discarding all terms that do not depend on w_n,

$$\min_{w_n} \quad \frac{1}{2}S_n||w_n||^2 - S_n w_n^\top m + C\sum_{i=1}^{N}(1 - y_{in}w_n^\top x_i)_+$$

where S_n denotes the number of neighbors for node n and m denotes the mean of neighbors $m = \frac{1}{S_n}\sum_{j:(n,j)\in E} w_j$. We now add a constant term $\frac{1}{2}S_n||m||^2$ (constant w.r.t this optimization subproblem) and divide by constant S_n and rewrite the optimization problem as,

$$\min_{w_n} \quad \frac{1}{2}||w_n - m||^2 + \frac{C}{S_n}\sum_{i=1}^{N}(1 - y_{in}w_n^\top x_i)_+ \tag{3.7}$$

Now, we solve (3.7) using the same coordinate descent technique as described earlier and recover the optimal w_n.

3.3.2 RR-LR

We follow a similar iterative strategy for optimizing RR-LR, i.e. we update the parameter w_n of each node n iteratively by fixing the rest of the parameters. Unlike RR-SVM, the objective function in RR-LR is convex and differentiable, therefore we can directly use quasi newton methods such as LBFGS for each inner optimization.

The update for each non-leaf node is given in eq (3.3). For each leaf node n, isolating the terms that depend on w_n, the objective and the corresponding gradient G can be written as

$$\min_{w_n} \quad \frac{1}{2}||w_n - w_{\pi(n)}||^2 + C\sum_{i=1}^{M}\log(1 + \exp(-y_{in}w_n^\top x_i)) \tag{3.8}$$

$$G = w_n - w_{\pi(n)} - C\sum_{i=1}^{M}\frac{1}{1 + \exp(y_{in}w_n^\top x_i)}y_{in}x_i \tag{3.9}$$

In the case of graphs, the only change is that instead of a single parent node, we subtract the difference from each neighbor,

$$G = \sum_{j:(n,j)\in E}(w_n - w_j) - C\sum_{i=1}^{M}\frac{1}{1 + \exp(y_{in}w_n^\top x_i)}y_{in}x_i \tag{3.10}$$

Note that although HBLR and RR-LR seem like two very different models, RR-LR can be viewed a successively simplified version of HBLR. If we replace the multiclass logistic function in HBLR

with multiple binary logistic functions and have a single common fixed α for all the nodes without any gamma prior, the HBLR model becomes identical to the RR-LR model - the only difference being the inference. Furthermore, if we estimate the parameters of HBLR using the posterior mode i.e MAP estimate instead of the posterior mean, even the training procedures become identical.

These connections also highlight the fundamental theme in all our proposed HBLR, RR-SVM and RR-LR models i.e. to enforce similarity between model parameters based on the hierarchical or graphical dependencies between the class labels. In the HBLR models, we enforce it by propagating a Bayesian prior over the hierarchical structure whereas in RR models we enforce it via regularization of parameters over the hierarchical or graphical structure.

3.3.3 Parallelization

On hierarchies, we can parallelize the optimization of the parameters exactly as discussed in section 2.4.3. For graphs however, parallelization is a little tricky. The **ideal** parallelization on graphs involves finding the chromatic number of the graph - which is the smallest K such that each node of the graph is assigned one of K different colors from 1 to K and no two adjacent nodes have the same color. Once the nodes have been assigned colors, we can repeatedly pick a color and parallely optimize the parameters of the nodes which have been assigned that color. However, finding the chromatic number of the graph is a NP-hard problem, therefore, we can only resort to approximate schemes to find the chromatic number. The degree of parallelization in graphs is given by $\frac{|\mathcal{N}|}{K}$ which is in contrast to $\frac{|\mathcal{N}|}{2}$ for hierarchies. Note that the minimum coloring needs to be solved only to achieve the best possible parallelization; in practice any *reasonable* greedy coloring achieves good parallelization.

Alternatively, one could also resort to other schemes such as performing multiple iterations of optimizing all nodes in parallel (although convergence is not guaranteed in theory). Furthermore, to aid in convergence, we also used other tricks such as warm starting with the previously found dual solution and random permutation of subproblems in coordinate descent method [Hsieh et al., 2008].

3.4 Experimental Setting

3.4.1 Datasets

We used the datasets outlined in section 2.6.1 (**CLEF**, **NEWS20**, **IPC**, **LSHTC-small**, **DMOZ-2010**) along with four other additional large-scale benchmark datasets. All dataset statistics

Table 3.1: Dataset Statistics

Dataset	#Training	#Testing	#Class-Labels	#Leaf-labels	Depth	#Features	Avg #labels per instance
CLEF	10,000	1,006	87	63	4	89	1
NEWS20	11260	7505	27	20	3	53975	1
RCV1	23,149	784,446	137	101	6	48,734	3.18
IPC	46,324	28,926	552	451	4	541,869	1
LSHTC-small	4,463	1,858	1,563	1,139	6	51,033	1
DMOZ-2010	128,710	34,880	15,358	12,294	6	381,580	1
DMOZ-2012	383,408	103,435	13,347	11,947	6	348,548	1
DMOZ-2011	394,756	104,263	35,448	27,875	6	594,158	1.03
SWIKI-2011	456,886	81,262	50,312	36,504	11	346,299	1.85
LWIKI	2,365,436	452,167	478,020	325,056	-	1,617,899	3.26

are tabulated in Table 3.1. To maintain comparability with previously published evaluations, we used the conventional train-test splits where-ever available.

1. **RCV1** [Lewis et al., 2004] A collection of Reuters News from 1996-1997. We used the topic-based classification as it has been most popular in evaluations.

2. **LSHTC-small**, **DMOZ-2010**, **DMOZ-2012** and **DMOZ-2011** Multiple web-page collections released as a part of the LSHTC (Large-Scale Hierarchical Text Classification) evaluation during 2010-12 [3]. It is essentially a subset of the web pages from the Open Directory Project.

3. **SWIKI-2011, LWIKI** Two subsets (small and large, respectively) of Wikipedia pages with human-generated topic class-labels. The dependencies between the class labels in SWIKI-2011 are given as links in a directed acyclic graph while in LWIKI they are given as links in a undirected graph.

Note that RCV1, DMOZ-2011, SWIKI-2011, LWIKI are multi-label datasets, meaning that an instance may have multiple correct labels; the other datasets only have one correct label per instance. For all the text datasets, we did the standard preprocessing steps of removing stop words, stemming followed by Tf-Idf (*ltc*) term weighting [Manning et al., 2008].

[3] http://lshtc.iit.demokritos.gr/node/3

3.4.2 Methods for Comparison

For evaluations, we compare the following methods

1. **RR models**: Our proposed methods, i.e., RR-SVM and RR-LR;
2. **HBLR models**: The Hierarchical Bayesian models developed in chapter 2 using partial MAP inference - M1-map, M2-map and M3-map;
3. **Flat baselines**: One-versus-rest classifiers BSVM, BLR and multiclass classifiers MSVM and MLR (described in section 2.6.2).
4. **Hierarchical baselines**: We used the hierarchical baselines described in section 2.6.2.

For all the above methods we tuned the regularization parameter using cross-validation with a range of values from 10^{-3} to 10^{3}. To scale up to the larger datasets, for the HBLR and RR models we used the approximate parallelization as discussed in sections 2.4.3, 3.3.3. The parallelization for the flat one-versus rest baselines (BSVM and BLR) is straightforward, i.e., simply learn the models for all the class-labels in parallel. Among the hierarchical baselines, only TD can be easily parallelized as the class models can be trained independently. It is not known how to parallelize the rest of the methods - MSVM, HSVM and OT and hence they cannot be scaled to the larger datasets. Therefore, we only report the results of these methods on the smaller datasets where they scaled.

On the multi-label datasets, to make the baselines as competitive as possible, we used an instance-based cut-off strategy as used in [Gopal and Yang, 2010]. This provided a better performance than using the usual cut-off of zero as well as other threshold methods like *rcut* or *scut* [Yang, 2001].

Note that HSVM, OT, MLR, MSVM and HBLR (with soft-max logistic function) are inherently multiclass methods and are not applicable in multilabeled scenarios. To enable HBLR models to be applicable on multilabel datasets, we replace the soft-max function with multiple binary logistic functions and use the same instance-based cut-off strategy. The prior parameters for HBLR was set as discussed in 2.5; to enable faster convergence on the larger datasets we estimated $\hat{p}$ in eq (2.13) using the parameters learnt by BLR.

3.5 Results

We evaluate the results using standard classification metrics - Micro-F_1 and Macro-F_1 (described in section B). We present comparisons with hierarchical and flat baselines as well well established results on large-scale benchmark datasets.

Table 3.2: **Comparison against flat baselines**: Macro-F_1 and Micro-F_1 on 10 datasets. Bold faced number indicates best performing method. NS denotes method does not scale to the dataset, NA denotes the method is not applicable since the dataset is multilabeled or has graph-based dependencies. The significance-test results between RR-SVM and BSVM;RR-LR and BLR on the first five datasets are denoted *for a p-value less than 5% and †for p-value less than 1%.

	BSVM	**BLR**	**MSVM**	**MLR**	**RR-SVM**	**RR-LR**	**HBLR** (M3-map)
CLEF							
Macro-F_1	48.59	53.26	54.33	54.76	53.92†	55.83*	**59.65**
Micro-F_1	77.53	79.92	80.02	80.52	80.02†	80.12*	**81.41**
RCV1							
Macro-F_1	54.72	53.38	NA	NA	56.56†	55.81*	**56.08**
Micro-F_1	80.82	80.08			81.66†	81.23†	**81.98**
NEWS20							
Macro-F_1	**82.32**	82.17	81.73	81.82	82.00	82.06	81.69
Micro-F_1	**83.10**	82.97	82.47	82.56	82.78	82.86	82.56
IPC							
Macro-F_1	45.71	48.29	NS	NS	47.89*	49.60	**51.06**
Micro-F_1	53.12	55.03			54.26†	55.37*	**56.02**
LSHTC-small							
Macro-F_1	28.62	28.12	28.34	28.38	28.94	28.48	**30.81**
Micro-F_1	45.21	44.94	45.62	45.20	45.31	45.11	**46.03**
DMOZ-2010							
Macro-F_1	32.64	31.58	NS	NS	**33.12**	32.42	31.88
Micro-F_1	45.36	45.40			**46.02**	45.84	45.64
DMOZ-2012							
Macro-F_1	31.59	14.18	NS	NS	**33.05**	20.04	27.19
Micro-F_1	56.44	52.79			**57.17**	53.18	53.15
DMOZ-2011							
Macro-F_1	24.34	21.67	NA	NA	**25.69**	23.90	23.46
Micro-F_1	42.88	41.29			**43.73**	42.27	41.35
SWIKI-2011							
Macro-F_1	26.57	19.51	NA	NA	**28.72**	24.26	NA
Micro-F_1	40.87	37.65			**41.79**	40.99	
LWIKI							
Macro-F_1	19.89	18.65	NA	NA	**22.31**	20.22	NA
Micro-F_1	37.66	36.96			**38.08**	37.67	

3.5.1 Comparison against Flat Baselines

Table 3.2 reports the results of our proposed models and against the flat baselines - BSVM, MSVM, BLR and MLR. The former two baselines are based on hinge-loss while the latter two methods are based on the logistic loss. On all the datasets except NEWS20, our proposed models perform better. To validate the results, we conducted pairwise significance tests between the RR and non-RR counterparts, i.e. between SVM and RR-SVM; LR and RR-LR. We used the sign test for Micro-F1 and wilcoxon rank test for Macro-F1. We show the results on the CLEF, IPC, LSHTC-small, NEWS20 and RCV1 datasets. We are unable to conduct significance tests on the other datasets since we did not have access to class-wise performance scores and true-test labels - our reported evaluation measures on these datasets are from the output of an online evaluation system which does not reveal classwise performance measures nor true test labels. The results of the significance tests on the five datasets show that the RR-models significantly outperform the non-RR models on three out of the five tested datasets.

Comparing the HBLR and RR models, on the smaller datasets like CLEF, RCV1, NEWS20 and IPC, the HBLR model M3-map offers the best performance, while on the larger datasets like DMOZ and Wikipedia, RR-SVM gives the best performance. This observation suggests that the choice of loss function affects the performance; hinge-loss seems to work better on the larger datasets (with high dimensions and skewed class-label distributions), while logistic loss is suited to datasets with balanced class distributions. Between HBLR and RR-LR, HBLR works better because it is a more expressive model than RR-LR. Moreover, we also see some correlation between the performance of HBLR and RR-LR; whenever HBLR performs best, RR-LR is second-best.

3.5.2 Comparison against Hierarchical Baselines

Table 3.3 compares the performance of our proposed methods against three hierarchical baselines - TD, HSVM and OT. On all the datasets, the proposed methods are able to leverage the hierarchy better and outperform existing hierarchical methods. On some datasets like LSHTC-small, there is a 16% relative improvement in performance. We conducted pairwise significance tests between the most scalable hierarchical baseline TD against our proposed methods - HBLR and RR. We used the setup as described in the section 2.7.1, sign test for Micro-F_1 and wilcoxon rank test for Macro-F_1. All the results are statistically significant.

3.5.3 Comparison against Benchmarks

Table 3.4 compares the results of our proposed models with the well established results on the large-scale datasets released by the LSHTC community. We focus on only those datasets for

Table 3.3: **Comparison against hierarchical baselines**: Macro-F_1 and Micro-F_1 on 10 datasets. Bold faced number indicates best performing method. NS denotes method does not scale to the dataset, NA denotes the method is not applicable since the dataset is multilabeled or has graph-base dependencies. The significance test results are between RR-SVM, RR-LR and HBLR against the scalable hierarchical baseline - TD on the first five datasets (†denotes significance at 1% level)

	TD	HSVM	OT	RR-SVM	RR-LR	HBLR (M3-map)
CLEF						
Macro-F_1	32.32	57.23	37.12	53.92†	55.83†	**59.65**†
Micro-F_1	70.11	79.72	73.84	80.02†	80.12†	**81.41**†
RCV1						
Macro-F_1	34.15	NA	NA	56.56†	55.81†	**56.08**†
Micro-F_1	71.34			81.66†	81.23†	**81.98**†
NEWS20						
Macro-F_1	80.86	80.04	81.20	82.00†	**82.06**†	81.69†
Micro-F_1	81.20	80.79	81.98†	82.78†	**82.86**†	82.56†
IPC						
Macro-F_1	42.62	NS	NS	47.89†	49.60†	**51.06**†
Micro-F_1	50.34			54.26†	55.37†	**56.02**†
LSHTC-small						
Macro-F_1	20.01	21.95	19.45	28.94†	28.48†	**30.81**†
Micro-F_1	38.48	39.66	37.12	45.31†	45.11†	**46.03**†
DMOZ-2010						
Macro-F_1	22.30	NS	NS	**33.12**	32.42	31.88
Micro-F_1	38.64			**46.02**	45.84	45.64
DMOZ-2012						
Macro-F_1	30.01	NS	NS	**33.05**	20.04	27.19
Micro-F_1	55.14			**57.17**	53.18	53.15
DMOZ-2011						
Macro-F_1	21.07	NA	NA	25.69	23.90	23.46
Micro-F_1	35.91			**43.73**	42.27	41.35
SWIKI-2011						
Macro-F_1	17.39	NA	NA	**28.72**	24.26	NA
Micro-F_1	36.65			**41.79**	40.99	

Table 3.4: **Comparison against established benchmark results** Macro-F_1 and Micro-F_1 of benchmark results established by the LSHTC challenge (we excluded our own submissions to the system). Bold faced number indicates best performing method. NA denotes that the method is not applicable on that dataset due to graph based dependencies between class labels.

	LSHTC Published Results	**RR-SVM**	**RR-LR**	**HBLR** (M3-map)
DMOZ-2010				
Macro-F_1	**34.12**	33.12	32.42	31.88
Micro-F_1	**46.76**	46.02	45.84	45.64
DMOZ-2012				
Macro-F_1	31.36	**33.05**	20.04	27.19
Micro-F_1	51.98	**57.17**	53.18	53.15
DMOZ-2011				
Macro-F_1	**26.48**	25.69	23.90	23.46
Micro-F_1	38.85	**43.73**	42.27	41.35
SWIKI-2011				
Macro-F_1	23.16	**28.72**	24.26	NA
Micro-F_1	37.39	**41.79**	40.99	
LWIKI				
Macro-F_1	18.68	**22.31**	20.22	NA
Micro-F_1	34.67	**38.08**	37.67	

which benchmark evaluations were available on the website[4]. Table 3.4 shows that the our proposed models are able to perform better than the state-of-the-art results reported so far on most of these datasets. In fact, on four out of the five datasets, RR-SVM shows a consistent 10% relative improvement than the currently published results.

3.6 Further Experimental Analysis

3.6.1 Efficiency Analysis

Table 3.5 reports the training times taken for all the methods. Among the flat baselines, the multiclass classifiers - MLR and MSVM cannot even be scaled to datasets with a moderate number of class labels. The one-versus-rest baselines on the other hand are very scalable and

[4]http://lshtc.iit.demokritos.gr/

3.6. Further Experimental Analysis

Table 3.5: **Computational efficiency**: The training time (in mins) for all the methods. NA denotes that the method is not applicable and NS denotes that the method is not scalable on that dataset.

	Flat baselines				Hierarchical Baselines			Proposed methods		
	BSVM	**BLR**	**MSVM**	**MLR**	**TD**	**HSVM**	**OT**	**RR-SVM**	**RR-LR**	**HBLR** (M3-map)
CLEF	.15	.24	.20	1.92	.13	3.19	1.31	.42	1.02	3.05
RCV1	.273	2.89	NA	NA	.213	NA	NA	.55	11.74	12.11
NEWS20	.04	.11	.07	.352	.04	2.31	.98	.14	.52	1.06
IPC	3.12	4.17	NS	NS	2.21	NS	NS	6.81	15.91	31.2
LSHTC-small	.31	1.93	1.65	3.63	.11	289.60	132.34	.52	3.73	5.2
DMOZ-2010	5.12	97.24	NA	NA	3.97	NS	NS	8.23	123.22	151.67
DMOZ-2012	22.31	95.38	NA	NA	12.49	NS	NS	36.66	229.73	331.96
DMOZ-2011	39.12	101.26	NA	NA	16.39	NA	NA	58.31	248.07	224.33
SWIKI-2011	54	99.46	NA	NA	21.34	NA	NA	89.23	296.87	NA
LWIKI	1114.23	2134.46	NA	NA	NA	NA	NA	2230.54	7282.09	NA

faster than our proposed models. BSVM is on average 1.92x faster than RR-SVM, BLR is on average 2.87x faster than RR-LR, and BLR is on average 4.89x faster than M3-map. This is not suprising - the better performance of our models comes at the cost of increased computational time. However even on the largest dataset LWIKI, RR-SVM takes about 37 hours, although slower than BSVM, the computation time certainly falls within the tractable range.

Among the hierarchical baselines, TD is the only one that can be scaled to the larger datasets, although with a low performance. HSVM and OT could only scale to the smallest datasets (CLEF and LSHTC-small) and both of them are orders of magnitude slower than our proposed methods. On the rest of the datasets neither of them could be tested succesfully either due to scalability issues or modelling (inability to handle multilabel data or graph-based dependencies) issues.

Between HBLR and RR-LR there is an efficiency vs effectiveness tradeoff. Although HBLR achieves a better performance than RR-LR with a 7% and $.5\%$ improvement in Macro-F_1 and Micro-F_1, it is 1.62x slower than RR-LR. The choice of the method depends on the user's preference between performance and scalability.

Table 3.6: The Macro-F_1 of BSVM, RR-SVM at various levels of the hierarchy along with the number of nodes at each level. The improvement in percentage of RR-SVM over BSVM is shown in parenthesis.

Level	**BSVM**	**RR-SVM**		**# Nodes**
	Macro-F_1	Macro-F_1	Improvement in Macro-F_1	
1	100.0	100.0	(0.00%)	1
2	71.43	**72.50**	(1.50%)	11
3	41.62	**42.93**	(3.12%)	342
4	25.71	**26.73**	(3.93%)	3376
5	26.54	**27.27**	(3.03%)	8858
6	13.06	**16.72**	(28.08%)	549

3.6.2 Performance across Training-set Sizes and Levels of Hierarchy

We present detailed analysis of the results of RR-SVM on two different fronts (a) the performance improvement at various levels of the hierarchy and (b) the performance improvement on across classes with varying number of positive training instances. Due to the lack of test-set labels (see section 3.5.1), we partitioned the training data from one of the datasets (DMOZ-2012) with a 50%-50% train-test split.

Table 3.6 reports the Macro-averaged F_1-score at various levels of the hierarchy. On all the levels our proposed RR-SVM performs better than SVM. The improvement seems to be particularly high at the lower levels (4, 5 and 6) where the leaf nodes are located.

Table 3.7 reports the Macro-averaged F_1-score for classes with different training set sizes. The improvement is highest in classes with moderately small number of positive training instances (6-50 training instances). The improvement seems to be smaller when the number of training instances is too low (1-5) or higher ($>$ 100). This is expected because, in the former case, learning a good classifier is generally very hard, and in the latter case, the amount of training data is already adequate and the hierarchy does not help further.

3.7 Summary

In this chapter, we proposed a unified framework that can leverage hierarchical and graphical dependencies between class-labels for improving classification. Our proposed two methods RR-SVM and RR-LR, rely on enforcing similarity between the model parameters based on the

Table 3.7: The Macro-F_1 of BSVM, RR-SVM across classes with different training set sizes. The number of classes in each range of training set size is also shown. The improvement in percentage of RR-SVM over BSVM is shown in parenthesis.

# Positive Instances	**BSVM** Macro-F_1	**RR-SVM** Macro-F_1	Improvement in Macro-F_1	**# Classes**
1-5	20.24	**20.62**	(1.86%)	7952
6-10	24.84	**26.59**	(6.57%)	1476
11-20	32.16	**34.08**	(5.63%)	1089
21-50	39.94	**41.23**	(3.12%)	680
51-100	49.25	**50.26**	(1.99%)	262
>100	59.81	**60.64**	(1.38%)	278

proximity of the classes in the provided structure. For improving the scalability of training, we developed fast and efficient parallel iterative optimization procedures.

Our proposed models achieved state-of-the-art results on multiple benchmark datasets and showed a consistent improvement in performance over both flat as well as other hierarchical methods.

Large-scale Multiclass Logistic Regression

4

4.1 Introduction

In this chapter, we develop a distributed approach to train Regularized Multinomial Logistic Regression (RMLR) model that can significantly increase the scalability of training when faced with very large number of classes and high dimensions.

RMLR is one of the fundamental tools to model the the probability of occurrence of one of many discrete outcomes or class-labels. With the increasing availability of data, there is growing interest in enabling such fundamental models to cope up with large-scale factors such as large number of class-labels and high dimensionality of the input space. To quantify the scale of data we are interested in, let us consider the DMOZ-2010 dataset studied in the previous chapter (section 3.4.1). It is a web-scale data collection with about 100,000 webpages organized into one of approximately 12,000 classes and each instance is represented by a sparse vector of 350,00 word-level features. To train a RMLR model, one would need to learn approximately $12000 \times 350000 = 4.2$ billion parameters which close to 17GB of parameters. Even if we overcome the hurdle of storing such large number of parameters in memory, it would still take significant amount of time to even perform a single iteration over all parameters. It is practically infeasible to learn a RMLR model with such large number of parameter on a single computer. In this chapter, we will address the issue of how to cope up with the computational and memory requirements of training RMLR models under such large-scale scenarios.

We propose a distributed approach to train RMLR models where the main chunk of the computation is accomplished by optimizing sets of parameters parallely across different computing units thereby gaining two big advantages (a) Faster computation - we can use the resources from multiple computing units to solve different parts of the same optimization problem (b) Distributed Memory - parameters can now be spread across multiple computing units simulta-

neously. Our approach relies on replacing the log partition function of the multinomial logit with a convex and parallelizable upper-bound based on the concavity of the log function. We prove that our new convex upper bound shares the same minimum as the original RMLR objective. We also explore several existing bounds to the log-partition function and discuss the applicability to the problem at hand. We additionally formulate an alternating direction method of multipliers (ADMM) [Boyd et al., 2011] version of the RMLR objective, a relatively recent technique where redundant constraints are added to the model to enable parallelization. We tested all the approaches on multiple datasets and found that our proposed convex upper bound offers the best trade-off between convergence and training time when the number of classes is large. To our knowledge, this is the first work that shows that RMLR models can be scaled to datasets with tens of thousands of class-labels and billions of parameters in a matter of several hours.

4.2 Related Work

To our knowledge, there is no directly related work that addresses the problem of training RMLR models for large number of classes, especially to the scale we are interested in. However we outline some of the most commonly used methods to train RMLR models and discuss the limitation of such models in large-scale settings.

The simplest approach to fit RMLR models are first order methods such as gradient descent. However, the problem with gradient methods is that they require careful tuning of the step-size and are generally slow to converge. Second order methods such as Newton's method improve over gradient methods in the sense that the descent direction provides a natural unit step-length and provide exponential convergence near the optimal solution. An efficient implementation of newton's method known as iterated reweighted least squares [Holland and Welsch, 1977] is widely used to fit logistic models. However second order methods fail to be useful in high dimensional settings as the calculation of descent direction involves a high dimensional matrix inversion (inverse of the Hessian) which is computationally intensive. Quasi newton methods such as BFGS [Shanno, 1985] and its limited memory variant LBFGS [Liu and Nocedal, 1989] are second order methods which overcome this problem of matrix inversion by continually updating an approximate inverse of the Hessian - although at a slower convergence rate. Recently, multiple studies [Sha and Pereira, 2003], [Schraudolph et al., 2007], [Daumé III, 2004] have shown that LBFGS methods offer the best trade-off between convergence and computational requirements and authors have often adopted it as the standard choice to train logistic models. However, in the large-scale setting we have at hand, even LBFGS does not meet the computational challenges. Firstly, LBFGS is not an inherently parallelizable (across parameters)

algorithm. Secondly, due to the requirement of repeated line-searches and function value evaluation, the parameters to be learnt typically have to be stored in memory (reading and writing 17GB of parameters of DMOZ-2010 to disk is expensive). Thirdly, LBFGS requires storing the gradient values from the last few iterations which further increase the memory requirements; for example if the gradient from the last 5 iterations is stored, the memory requirement shoots upto $17 \times 5 \sim 85$GB making it practically infeasible.

Other ways to train RMLR are using iterative scaling [Darroch and Ratcliff, 1972], conjugate gradient [Lin et al., 2008] and dual methods [Yu et al., 2011]. To our knowledge, there has been no work on large-scale distributed RMLR training using these methods particularly to the scale we are interested in. Also refer [Minka, 2003] for an excellent comparison between the different existing approaches for *binary* logistic regression.

Indirectly related to this work are a few works in stochastic gradient descent [Zhang, 2004], [Bottou, 2010] and online learning [Bottou, 1998]. Stochastic and online approaches address scalability issues when the number of training instances is large. This is very different from our goal where the focus is large number of classes rather than large number of training instances. Moreover, unlike stochastic approaches our aim is not to provide guarantees on expected (expectation w.r.t data) error of the model but training a RMLR model on a fixed dataset.

4.3 Parallel Training by Decoupling Log-Normalization Constant

Let $\{x_i, t_i\}_{i=1}^{i=N}$ denote the set of training instances where each $x_i \in \mathcal{R}^D$ and $t_i \in \{1, 2, ..K\}$ and K denotes the number of class labels. Define the indicator $y_{ik} = I(t_i = k)$ which denotes whether the i^{th} training example belongs to class k or not. Let the probability of a given instance x to have a class-label k be modelled as,

$$P(y = k|x) = \frac{exp(w_k^\top x)}{\sum_{k'=1}^{K} exp(w_{k'}^\top x)}$$

where $\mathbf{W} = \{w_1, w_2, ..w_K\}$ denote the set of parameters. The training objective of RMLR can be written as,

$$\textbf{[OPT 1]} \qquad \min_{\mathbf{W}} \; G_{\mathcal{D}}(W)$$

$$\text{where} \quad G_{\mathcal{D}}(W) = \frac{\lambda}{2}\sum_{k=1}^{K}\|w_k\|^2 - \sum_{k=1}^{K}\sum_{i=1}^{N} y_{ik} w_k^\top x_i + \sum_{i=1}^{N}\log\left(\sum_{k'=1}^{K}\exp(w_{k'}^\top x_i)\right)$$

The most natural way to parallelize is to optimize each class-level parameter w_k in parallel. However this is not directly possible due to the presence of the log partition function which couples all the class-level parameters together inside a log-sum-exp function. This makes the objective non decomposable across the w_k's. This leads to the question - can we replace the log-partition function by a *parallelizable* function ? Secondly, can this *parallelizable* function also be an upper-bound to the log-partition function as this would guarantee that true-minimum is at most as high as the minimum of the upper bounded objective. And finally, the introduction of this parallelizable function must not make the problem *harder* to solve (like for example make it non-differentiable or multiple local minima etc).

To this end, we explore 3 different bounds for the log partition function and their applicability to the problem at hand.

4.3.1 Piecewise Bound

One of the properties of convex functions is that they can be approximated by piece-wise linear functions to any degree of precision just by increasing the number of pieces. This property can be used to upper and lower-bound the log-sum-exp function [Hsiung et al., 2008]. The idea is to find a set of variational parameters a, b, c, d such that

$$\max_j\{a_j^\top\gamma + b_j\} \leq \log\left(\sum_{k=1}^{K}\exp(\gamma_k)\right) \leq \max_{j'}\{c_{j'}^\top\gamma + d_{j'}\}$$
$$a, c \in \mathcal{R}^K \quad b, d \in \mathcal{R}$$

A similar idea was also used in [Marlin et al., 2011] to approximate a binary logistic function but with a quadratic function instead of linear functions. However there a few problems associated with such piecewise bounds. Firstly, finding the variational parameters a, b, c, d is not easy. Till date a constructive solution for arbitrary precision for the log-sum-exp function has been established only for $K = 2$ [Hsiung et al., 2008]. Without a constructive solution one needs to resort to derivative free optimization methods like Nelder Mead method etc to fix the value of the variational parameters. More over the number of such variational parameters also grows linearly with the number of classes K, therefore for datasets with a large number

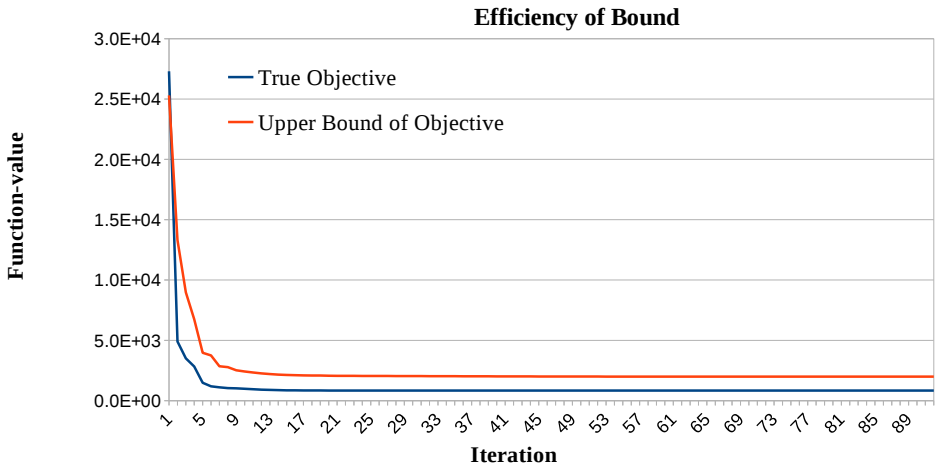


Figure 4.1: The figure shows the difference between the true minimum attained by the function and the upper bound using double majorization.

of classes, finding the variational parameters through derivative free methods would be even harder. Secondly, even if one were to find such parameters, the objective function is still not parallelizable across the class-level parameters. Thirdly, using this bound would introduce non-differentiability in the objective i.e. the log-sum-exp function would be replaced by a max over linear functions, thereby rendering the objective non-differentiable. Due to the above issues, piece-wise linear bounds do not help us parallelize the training procedure for RMLR models.

4.3.2 Double Majorization Bound

This bound was proposed in [Bouchard, 2007] to enable variational inference for logit models with Gaussian priors. The bound is given by,

$$\log\left(\sum_{i=1}^{K}\exp(w_k^\top x_i)\right) \leq a_i + \sum_{k=1}^{K}\log(1+e^{w_k^\top x_i - a_i}) \tag{4.1}$$

where the variational parameters $a_i \in \mathcal{R}$.

Firstly, note that the bound is parallelizable as it splits as sum of functions of class-wise parameters w_k and each w_k can be optimized as,

$$\arg\min_{w_k} \quad \frac{\lambda}{2}\|w_k\|^2 - \sum_{i=1}^{N} y_{ik} w_k^\top x_i + \sum_{i=1}^{N}\log(1+e^{w_k^\top x_i - a_i}) \tag{4.2}$$

Secondly, it is differentiable and thirdly and importantly the upper bound is still convex. Although this bound seems to possess all the properties we need, the problem is that the bound

is not tight enough. In our initial experiments we found that the gap between the log-sum-exp function and this upper bound is large. Figure 4.1 shows the gap between the true objective and the upper-bounded objective. The graph was generated by training two RMLR on the 20 news group dataset using the LBFGS optimization algorithm [Liu and Nocedal, 1989] and plotting the function-value after each iteration. The blue line shows the function-value using log-sum-exp and the red line shows the function-value by using the upper-bound. Since the gap is relatively large, we do not recommend using this bound.

4.3.3 Log-concavity Bound

This is a well known bound that exploits the first order concavity property of the log-function. It has been used in many works including [Blei and Lafferty, 2006a], [Bouchard, 2007]. The bound is as follows,

$$log(\gamma) \leq a\gamma - log(a) - 1 \quad \forall \gamma, a > 0 \tag{4.3}$$

where a is a variational parameter. Minimizing the RHS over a gives $a = \frac{1}{\gamma}$ and makes the bound tight. To incorporate this into the objective, we introduce variational parameters a_i for each training instance i. We have the log-partition function for instance i bounded as,

$$\log\left(\sum_{k=1}^{K} \exp(w_k^\top x_i)\right) \leq a_i \sum_{k=1}^{K} \exp(w_k^\top x_i) - log(a_i) - 1$$

Firstly, note that incorporating this into the objective makes the objective parallelizable across w_k's. That is, for a given fixed value of the variational parameter a_i, the optimization over $\mathbf{W}$ splits into a sum of K different objectives. Specifically, the class-level parameter for class k i.e. w_k can be estimated by solving,

$$\arg\min_{w_k} \quad \frac{\lambda}{2}\|w_k\|^2 - \sum_{i=1}^{N} y_{ik} w_k^\top x_i + \sum_{i=1}^{N} a_i \exp(w_k^\top x_i) \tag{4.4}$$

Secondly, note that finding the variational parameter is an easy problem because optimizing over a_i in (4.3) has a closed form solution. Thirdly note that the combined objective (as given below), is still differentiable.

$$\min_{\mathbf{A}>0,\mathbf{W}} \frac{\lambda}{2}\sum_{k=1}^{K}\|w_k\|^2 + \sum_{i=1}^{N}\left[-\sum_{k=1}^{K} y_{ik} w_k^\top x_i + a_i \sum_{k=1}^{K} \exp(w_k^\top x_i) - \log(a_i) - 1\right] \tag{4.5}$$

The only down-side of this bound is that the convexity of the original objective is now lost due to the presence of a product of linear and exponential function i.e. $a_i \exp(w_k^\top x_i)$. This

introduces the possibility of potentially multiple local minima in the objective and hence the upper-bound could be potentially loose. In our previous work [Gopal and Yang, 2013a] we show that these problems can be alleviated and develop a block coordinate descent approach that still converges to the optimal solution.

In this chapter, we take an alternative route, and show that a simple change of variables in (4.5) results in an unconstrained convex optimization problem which is also parallelizable and importantly shares the same minimum as the true RMLR objective. More specifically, by introducing $\xi_i = -log(a_i)$, we can rewrite (4.5) as the following optimization problem,

[OPT 2] $\min_{\boldsymbol{\xi},\mathbf{W}} F_{\mathcal{D}}(\mathbf{W},\boldsymbol{\xi})$

$$\text{where} \quad F_{\mathcal{D}}(\mathbf{W},\boldsymbol{\xi}) = \frac{\lambda}{2}\sum_{k=1}^{K}\|w_k\|^2 + \sum_{i=1}^{N}\left[-\sum_{k=1}^{K} y_{ik} w_k^\top x_i + \sum_{k=1}^{K}\exp(w_k^\top x_i - \xi_i) + \xi_i - 1\right]$$

Note that $F_{\mathcal{D}}(\mathbf{W},\boldsymbol{\xi})$ is a convex function; the non-convexity due to multiplication of a linear and a exponential is now an exponential of a linear function which is convex, all the other terms are also individually convex.

To show that we can recover the original RMLR solution from OPT2, consider the optimal solution $\{\mathbf{W}^*,\boldsymbol{\xi}^*\}$ of OPT2. By setting the partial gradients of each parameter to zero,

$$\xi_i^* = \log\left(\sum_{k=1}^{K}\exp({w_k^*}^\top x_i)\right)$$

$$\lambda\sum_{k=1}^{K} w_k^* + \sum_{i=1}^{N} x_i\left(\frac{\exp({w_k^*}^\top x_i)}{\exp(\xi_i^*)} - y_{ik}\right) = 0$$

It it easy to see that the optimality conditions of $\mathbf{W}$ above is identical to the optimality conditions for $G_{\mathcal{D}}(\mathbf{W})$ in OPT1 i.e. any optimal $\mathbf{W}$ of OPT2 is also an optimal solution to OPT1. Since OPT1 is strongly convex, there is exactly one optimal solution to OPT1. Therefore, OPT2 has exactly one optimal solution $\{\mathbf{W}^*,\boldsymbol{\xi}^*\}$ which is the same as OPT1 (note that $\boldsymbol{\xi}^*$ is uniquely determined by $\boldsymbol{W}^*$)

At this point we can use any off the shelf convex optimization routine to solve OPT2. However, since we are interested in parallelizing the objective across parameters, we use a simple block coordinate method to solve OPT2 (outlined in Algorithm 2). Note that this block coordinate descent is identical to the one in [Gopal and Yang, 2013a] and leads to the same updates.e

We can use the norm of the gradient as the convergence criteria. The norm of the gradient can be easily calculated between steps 1 and 2 of Algorithm 2 as follows,

$$\|\nabla F_{\mathcal{D}}(\boldsymbol{W}^{t+1},\boldsymbol{\xi}^t)\| = \left\|\sum_{i=1}^{N}\left[1 - \frac{1}{\exp(\xi_i^t)}\sum_{k=1}^{K}\exp(w_k^{(t+1)\top} x_i)\right]\right\|$$

Algorithm 2 Block coordinate descent algorithm

Initialize : $t \leftarrow 0, \boldsymbol{\xi}^0 \leftarrow log(K), \mathbf{W}^0 \leftarrow 0.$
Result : $\mathbf{W}^t$
While : Not converged
1. *In parallel* : $\mathbf{W}^{t+1} \leftarrow \arg\min_W F_{\mathcal{D}}(W, \boldsymbol{\xi}^t)$
2. *In parallel* $\boldsymbol{\xi}^{t+1} \leftarrow \arg\min_{\xi} F_{\mathcal{D}}(\mathbf{W}^{t+1}, \xi)$
3. $t \leftarrow t + 1$

Note that the objective $F_{\mathcal{D}}(\boldsymbol{W}, \boldsymbol{\xi})$ is strictly convex function, w.r.t $[\mathbf{W}, \boldsymbol{\xi}]$ as well strongly strongly convex w.r.t $\mathbf{W}$ with parameter λ (for a fixed $\boldsymbol{\xi}$). We also know that there exists a finite solution $\mathbf{W}^*, \boldsymbol{\xi}^*$, as well a unique solution for steps 2 and 3 in algorithm 2. Convergence for such methods has been established by [Tseng, 2001].

Regarding rates of convergence, previous works have established linear convergence rate for randomized coordinate descent [Nesterov, 2012],[Richtárik and Takáč, 2014]. Recently [Beck and Tetruashvili, 2013] have shown sublinear convergence rates for two-block coordinate minimization for smooth convex functions with lipschitz continuous gradients. Note that our objective $F_D(\mathbf{W}, \boldsymbol{\xi})$ does not have gradients which are globally lipschitz due to the presence of the exponential function but are however locally lipschitz. Convergence rates for complete minimization of blocks of coordinates for general convex function is still an active area of research.

4.4 Parallel Training by ADMM

Alternating direction method of multipliers (ADMM) is a relatively new technique that enables reformulating simple convex optimization problems in a manner that can be easily parallelized. The key idea in ADMM is to introduce redundant linear constraints into the problem such that the optimization of the objective can be parallelized. In [Boyd et al., 2011], the authors show how ADMM can enable distributed computing for many machine learning models by either splitting across examples or splitting across parameters. Since we are particulary interested in parallelizing across class-parameters, we will adapt the ADMM formulation for splitting across features. For a problem of the form

$$\min_{\mathbf{v}} \sum_{k=1}^{K} f_k(v_k) + g(\sum_{k=1}^{K} v_k)$$

the corresponding ADMM formulation is given by,

$$\min_{\mathbf{V},\mathbf{Z}} \sum_{k=1}^{K} f_k(v_k) + g(\sum_{k=1}^{K} z_k)$$
$$\text{subject to} \quad v_k - z_k = 0, \qquad k = 1, ..K.$$

Here v_k are the parameters in the original problem and the z_k are the additional parameters introduced to enable distributed computing. f and g are both convex functions. The optimization problem is first solved by optimizing for each of the v_k in parallel. In the second step, to solve for z, instead of solving the problem as an optimization problem over k variables, we reduce it to an optimization problem over a single variable $\bar{z}$ by re-writing g as $g(K\bar{z})$. This neat trick enables efficient parallelization (refer [Boyd et al., 2011] pgs 56-57, 67-68)

However, this technique does not given any benefit when applied to RMLR model. For simplicity, consider a RMLR formulation with just one example x and $\lambda = 0$. The optimization problem for RMLR is given by

$$\min_{\mathbf{W}} \quad -\sum_{k=1}^{K} y_k w_k^\top x_i + \log\left(\sum_{k=1}^{K} \exp(w_k^\top x)\right)$$

To formulate the corresponding ADMM problem, it is clear that v_k needs to be set to class-level parameter w_k and $f_K(w_k) = -y_k w_k^\top x_i$ and g refers to the log-sum-exp function. Next we introduce redundant variables z_k such that $z_k = w_k^\top x_i$ for $k = 1, ..K$ (note that the redundant constraints in ADMM should be linear). Now applying the ADMM formulation for N examples, we need to introduce $N \times K$ redundant variables i.e. z_{ik} variables which represent the prediction of training instance i w.r.t class k. This is given by,

$$\min_{\mathbf{W}} \quad -\sum_{k=1}^{K}\left(\sum_{i=1}^{N} y_k w_k^\top x_i\right) + \sum_{i=1}^{N} \text{log-sum-exp}(z_i)$$
$$\text{subject to} \quad w_k^\top x_i - z_{ik} = 0, \qquad i = 1, ..N \;\; k = 1, ..K.$$

For solving, in each iteration, we need to solve K optimization problems of D (dimension) variables (i.e. for each w_k) and N optimization of K variables each (i.e. for each $\{z_{ik}\}_{k=1}^{K}$). There are several problems in using this approach for training RMLR models. Firstly, the above requires much more computation than the log-concavity bound which requires solving only K optimization problems of D (dimension) variables (i.e. for each w_k) in each iteration. Secondly, the introduction of Z variable increases memory by $O(NK)$ as opposed to $O(N)$ variational parameters using the log-concavity bound . Lastly, as noted in [Boyd et al., 2011] (pg 17), ADMM exhibits very slow convergence properties. As we will see in the next section, it takes

orders of magnitudes more computation (even after parallelization) to reach the same accuracy of optimization as log-concavity bound or LBFGS.

4.5 Experiments

Throughout our experiments, we consider 4 different datasets with increasing number of parameters to learn - CLEF [1], NEWS-20 [2], LSHTC-small, DMOZ-2010 [3] (An outline of the various characteristics of the dataset is given in Table 3.1).

We compare the following methods for distributed training of RMLR models,

1. **ADMM:** The alternating direction method of multipliers discussed in section 4.4. We tried multiple values of the ρ parameter [Boyd et al., 2011] and chose the one that offered the fastest convergence.

2. **LC:** The log-concavity bound in section 4.3.3.

3. **LBFGS:** The standard quasi-newton methods widely used to train logistic models [Liu and Nocedal, 1989]. All the parameters of the model are optimized simultaneously. We use the previous 5 gradients to update the approximate hessian matrix. Furthermore, in order to make LBFGS as competitive as possible, the dot-product of an instance x_i with the class parameters w_k are computed in parallel (i.e. enables parallel computation of gradient).

4. **DM:** The Double Majorization in section 4.3.2.

For each inner problem in LC (4.4), DM (4.2) and ADMM we use LBFGS for optimization (other solvers can also be used, but LBFGS was chosen to maintain comparability). All methods were tested on a 48 core AMD Opteron Processor with 32GB RAM. For the largest DMOZ-2010 dataset, a map-reduce based Hadoop 20.2 cluster with 64 worker nodes with 8 cores and 16GB RAM (cluster has 220 mappers and 196 reducers) was used. Only the LC method could be scaled on this dataset. In each iteration of LC, the class parameters are parallely optimized in the map-phase and the variational parameters are updated in the reduce phase. Note that Hadoop in not a requirement, we chose to use it because the cluster could only be interfaced through Hadoop. Infact, other alternatives such as Peregrine [4], Haloop [5] , Twister [6] or non-hadoop alternatives such as MPI might be better choices as they can be customized

[1] http://kt.ijs.si/DragiKocev/PhD/resources/doku.php?id=hmc_classification
[2] http://people.csail.mit.edu/jrennie/20Newsgroups/
[3] http://lshtc.iit.demokritos.gr/
[4] http://peregrine_mapreduce.bitbucket.org/
[5] http//code.google.com/p/haloop/
[6] http://www.iterativemapreduce.org/

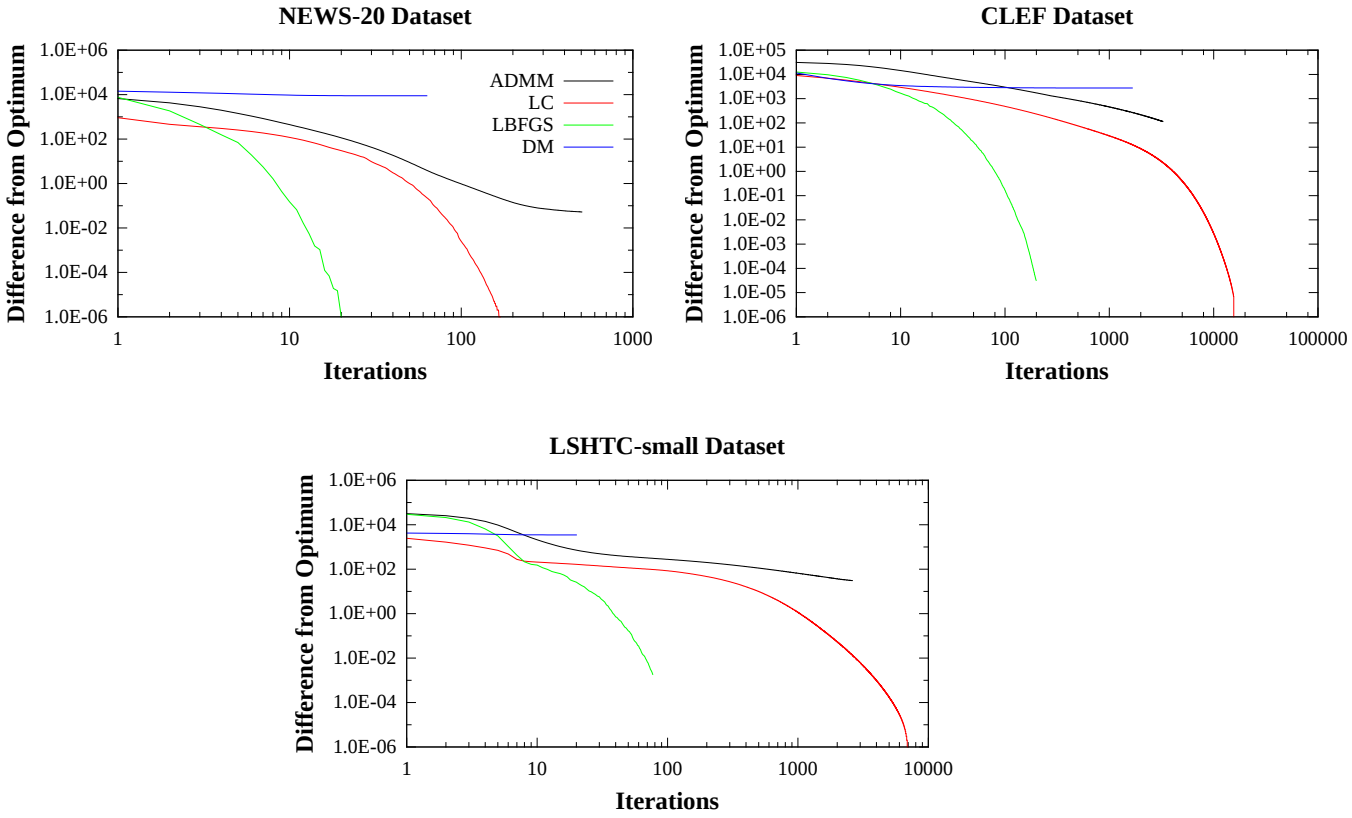


Figure 4.2: The difference from the true optimum as a function of number of iterations

for iterative parallelizable computations. The λ parameter for the datasets was selected using cross-validation using a one-versus rest logistic regression; for NEWS20 and CLEF, $\lambda = 1$, for LSHTC-small, $\lambda = .001$ and DMOZ-2010, $\lambda = .01$.

Figures 4.2 show the difference from the optimal solution as the number of iteration increases for the first three smaller datasets. In all the graphs, we see a common pattern : LBFGS takes the fewest number of iterations to converge to the optimal solution, followed by LC, ADMM and DM. This is not surprising because quasi-newton methods like LBFGS store some approximation of the Hessian which helps to achieve faster convergence in fewer steps than other methods. LC although being a block coordinate descent method seems to offer a much better convergence compared to ADMM or DM. ADMM as noted by the authors exhibits very slow convergence [Boyd et al., 2011] (pgs 6-7) to accurate solutions. DM does not even reach the optimal solution since the DM bound (4.1) is not tight.

Figures 4.3 show the difference from the optimal solution as a function of time taken. In this case, there is considerable shift in results. Among the four methods, only two of them

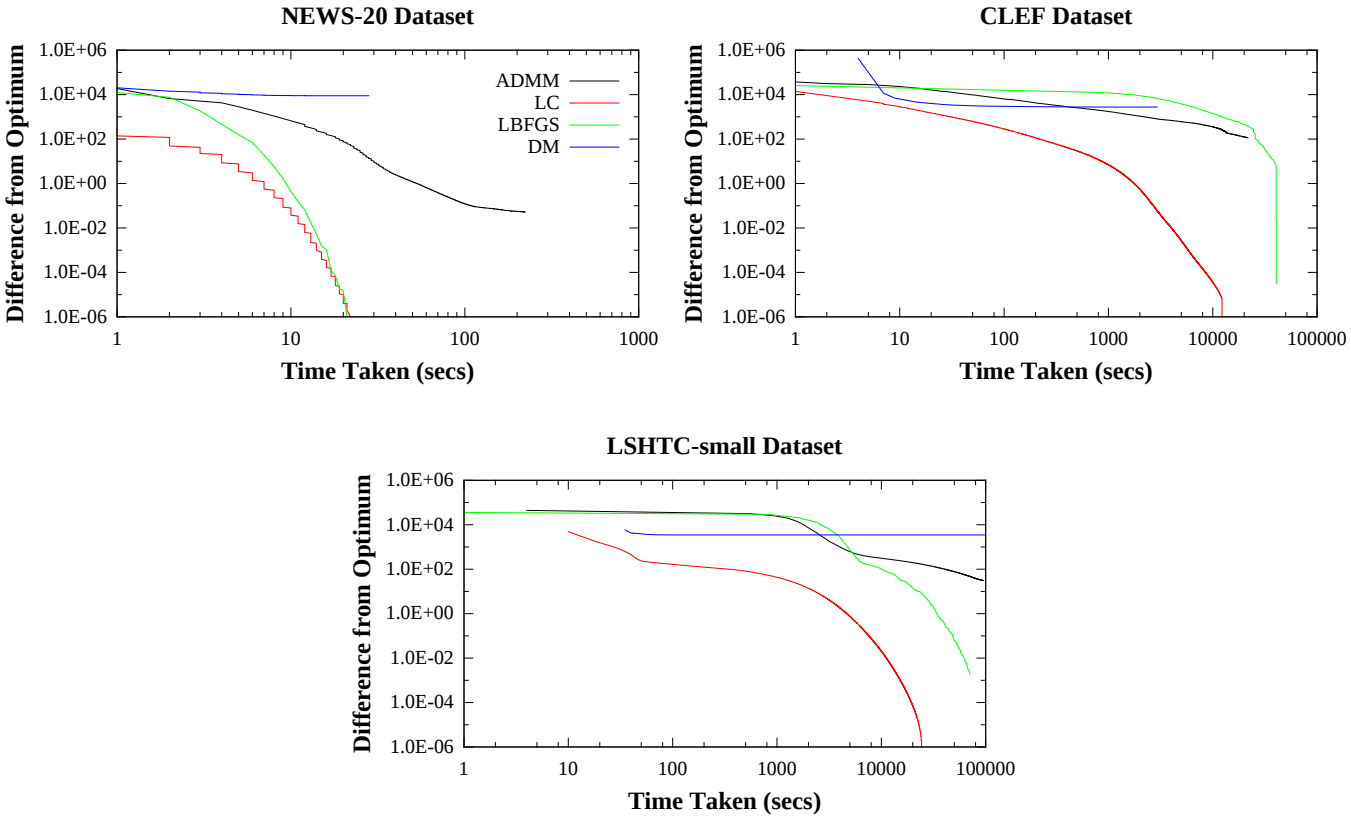


Figure 4.3: The difference from the true optimum as a function of time in seconds.

seem to show reasonable convergence with time - LBFGS and LC. Comparing the 2 methods on the smallest dataset NEWS20, there does not seem to much difference between the them. But as the number of classes gets larger like the CLEF and LSHTC-small datasets, LC seems to perform significantly faster than LBFGS (and the other methods). Although LBFGS takes fewer iterations, each iteration is very computationally intensive and takes a long time, therefore each step of LBFGS is time consuming. This in contrast to LC where the cost per iteration is very cheap since the parameters are optimized independently.

Infact on the largest DMOZ-2010 dataset, it is not even possible to run LBFGS due to the extreme memory requirements (17GB of parameters + 85GB of past gradient values need to be stored simultaneously in main memory). However, LC overcomes this difficulty by iteratively solving multiple subproblems and distributing the parameters across several computing units. Figure 4.4 shows the progress of LC on DMOZ-2010 as the number of iterations/time progresses. Most iterations of LC takes less than 6 minutes: around 3-4 minutes for optimizing the class-parameters and 2 minutes to update the variational parameters. This includes the time taken

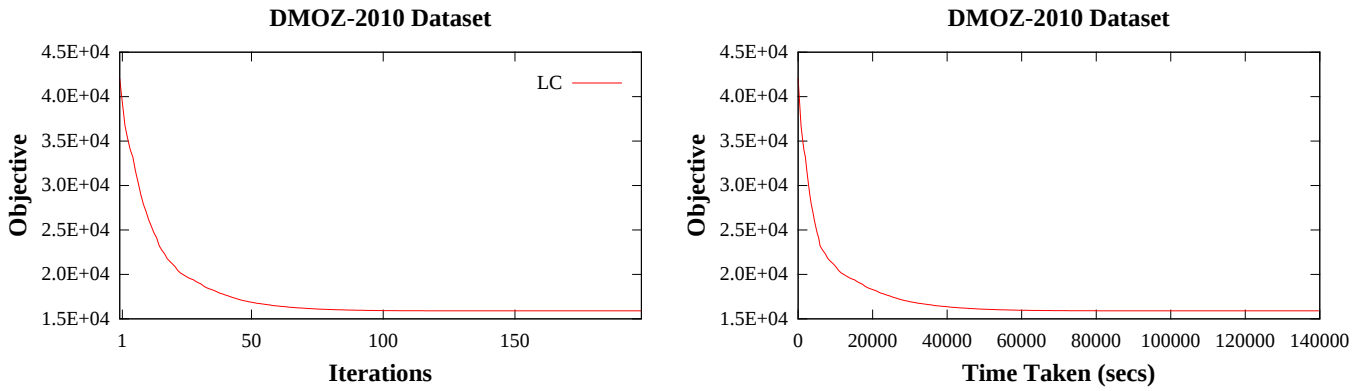


Figure 4.4: The progress of LC on the DMOZ-2010 dataset

to start the hadoop job and transfer the parameters etc. Note that it is possible to train RMLR using ADMM or DM on this dataset, but we did not pursue this since neither of them showed reasonable convergence on the other datasets.

4.6 Summary

In this chapter, we developed four ways to parallelize the training of RMLR models. Our analysis and experiments establish that on datasets with larger number of parameters, ADMM offers very slow convergence rate, DM has a loose upper bound, LBFGS is too memory intensive and LC being the most effective method. Unlike other methods, LC is able to succesfully exploit parallelization and achieve $\sim$10x speedup in training time.

Clustering on Unit-spheres with von Mises-Fisher Mixtures

5

5.1 Introduction

Unit normalized data representation is a standard way of representing data in many fields such as information retrieval, text classification [Salton and McGill, 1986], image classification etc, and has often achieved better performance than typical bag-of-counts representation [Robertson, 2004],[Joachims, 2002]. Existing popular methods for clustering such as mixture of Multinomial distributions, Latent Dirichlet Allocation [Blei et al., 2003] (LDA) with its variants and the popular Gaussian mixtures, although very flexible, are not suitable for modelling such unit normed data. The former methods cannot even be applied on continuous real values while the latter Gaussian distribution wastes probability mass in modelling instances outside the unit sphere domain.

In this chapter, we develop a suite of clustering models that retain the flexibility of existing graphical models but at the same time are particularly suited to unit normalized data. Our models are based on the von Mises-Fisher distribution (a probability density on unit spheres) and have the ability to extract a flat set of clusters, a hierarchically coherent set of clusters, or time-varying clusters from the data. More specifically, we propose the following models,

1. *The full Bayesian vMF Mixture Model* (**B-vMFmix**) - a mixture model based on the von Mises-Fisher (vMF) distribution for clustering data on unit sphere. The Bayesian model describes a generative process where each instance is drawn from one of many vMF distributions (clusters) and the parameters of the clusters are themselves drawn from a common prior which helps the clusters to share information among each other.

2. *The Hierarchical vMF Mixture Model* (**H-vMFmix**): When the data we want to analyze is huge, one of the efficient means of browsing is by means of a hierarchy [Cutting et al.,

1992]. We extend B-vMFmix to H-vMFmix, to enable partitioning the data into increasing levels of specificity as defined by a given input hierarchy. To our knowledge, this is the first hierarchical Bayesian model for vMF-based clustering.

3. *The Temporal vMF Mixture Model* (**T-vMFmix**): For temporal data streams, analyzing how the latent clusters in the data evolve over time is naturally desirable. We augment B-vMFmix to the first temporal vMF-based model that accommodates changes in cluster parameters between adjacent time-points. For example, in a corpus of documents, this could reflect the changing vocabulary within a cluster of documents.

The key difficulty that we address in these models is in developing efficient numerical techniques that can estimate the resulting posterior distribution of the model parameters. Specifically, we present three methods for posterior estimation,

- A full variational inference algorithm.
- A partial variational inference with sampling step.
- A Collapsed Gibbs sampling algorithm.

We conducted thorough experiments on a wide variety of text data and evaluated the clustering models on two fronts - *Recoverability*, the ability to generate clusters that match the ground-truth and *Generalizability*, the ability to generalize well to unseen test data. In terms of *recoverability* we found that vMF based models significantly outperform other popular non-vMF clustering models including mixture of Multinomials, K-means, LDA etc. In terms of *generalizability* we demonstrate the ability of our models to fit better to unseen data than existing vMF based models.

5.2 Related Work

The von Mises-Fisher (vMF) distribution defines a probability density over points on a unit-sphere. It is parameterized by mean parameter μ and concentration parameter κ - the former defines the direction of the mean and the latter determines the spread of the probability mass around the mean. The density function for $x \in \mathcal{R}^D, \|x\| = 1$ is,

$$f(x|\mu, \kappa) = C_D(\kappa) \exp(\kappa \mu^\top x) \quad \text{where} \quad C_D(\kappa) = \frac{\kappa^{.5D-1}}{(2\pi)^{.5D} \mathcal{I}_{.5D-1}(\kappa)}, \ \|\mu\| = 1, \ \kappa > 0$$

where $\mathcal{I}_\nu(a)$ is the modified bessel function of first kind with order ν and argument a. Note that $\mu^\top x$ is the cosine similarity between x and mean μ and that κ plays the role of the inverse of variance.

5.2. Related Work

vMF distributions have been long studied in the directional statistics community [Fisher, 1953, Jupp and Mardia, 1989]. They model distances between instances using the angle of separation i.e. cosine similarity. Early work in vMF focused on low-dimensional problems (2D or 3D spaces) to maintain tractability and relied on auxiliary Gibbs sampling for inference which is generally difficult to scale in higher dimensions [Mardia and El-Atoum, 1976] [Guttorp and Lockhart, 1988] [Bangert et al., 2010]. More recent studies include spherical topic models [Reisinger et al., 2010] that mimics LDA with a Bayesian inference for learning the mean parameters in vMF but leave the crucial concentration parameters to be set manually and the text clustering work by [Banerjee et al., 2006] [Banerjee et al., 2003] where an EM-based algorithm without Bayesian inference was used for parameter estimation. In the simplest vMF mixture model as described in [Banerjee et al., 2006] each instance is assumed to be drawn from one of the K vMF distributions with a mixing distribution π, where K is a prespecified constant, and the parameters of the vMF distributions correspond to the underlying themes (clusters) in the data. The cluster assignment variable for instance x_i is denoted by $z_i \in \{1, 2, ..K\}$ in the probabilistic generative model given below,

$$\begin{aligned} z_i &\sim \text{Categorical}(.|\pi) \quad i = 1, 2..N \\ x_i &\sim \text{vMF}(.|\mu_{z_i}, \kappa) \quad i = 1, 2..N \end{aligned}$$

The k'th cluster is defined by mean parameter μ_k and concentration parameter κ. The parameters $\mathcal{P} = \{\boldsymbol{\mu}, \boldsymbol{\kappa}, \pi\}$ are treated as fixed unknown constants and $\mathbf{Z} = \{z_i\}_{i=1}^N$ are treated as a latent variables. To train the model, the familiar EM algorithm was used to iterate between calculating the $E[\mathbf{Z}]$ in the E-step and optimizing $\mathcal{P}$ to maximize the likelihood in the M-step. In the E-step, the expected cluster membership is,

$$E[z_{ik}] = \frac{\pi_k \text{vMF}(x_i|\mu_k, \kappa)}{\sum_{j=1}^{K} \pi_j \text{vMF}(x_i|\mu_j, \kappa)}$$

The M-step update of parameters is,

$$\pi_k = \sum_{i=1}^{N} E[z_{ik}] \quad , \quad R_k = \sum_{i=1}^{N} E[z_{ik}] x_i \quad , \quad \mu_k = \frac{R_k}{\|R_k\|}$$

The optimal κ lacks a closed form solution, but a very good approximation with a closed form expression was developed in [Banerjee et al., 2006],

$$\bar{r} = \frac{\sum_{k=1}^{K} \|R_k\|}{N} \quad , \quad \kappa = \frac{\bar{r}D - \bar{r}^3}{1 - \bar{r}^2}$$

5.3 Bayesian von Mises-Fisher Mixtures (B-vMFmix)

Given data $\mathbf{X} = \{x_i\}_{i=1}^N$ and a fixed number of clusters K, the generative model for the Bayesian von Mises-Fisher mixture model is as follows,

$$\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_k \sim \text{vMF}(.|\mu_0, C_0) \quad k = 1, 2, ..K \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \quad k = 1, 2, ..K \\
&z_i \sim \text{Categorical}(.|\pi) \quad i = 1, 2...N \\
&x_i \sim \text{vMF}(.|\mu_{z_i}, \kappa_{z_i}) \; i = 1, 2, ..N
\end{aligned}$$

In this model, each instance x_i is assigned to cluster z_i, which is drawn from a multinomial distribution over K clusters with a parameter π. The parameters of the k'th cluster are denoted by a vMF distribution with mean parameter μ_k and concentration parameter κ_k. The cluster mean parameters $\boldsymbol{\mu} = \{\mu_k\}_{k=1}^K$ are commonly drawn from a prior vMF distribution with parameters $\{\mu_0, C_0\}$. The mixing distribution π is drawn from a symmetric dirichlet with parameter α. The cluster concentration parameters $\boldsymbol{\kappa} = \{\kappa_k\}_{k=1}^K$ are commonly drawn from a log-normal prior with mean m and variance σ^2 (one could also choose a gamma prior, but lognormal was chosen due to the availability of more easily interpretable mean and variance parameter and also their closed form Empirical Bayes estimates). The prior parameters in this model are $\{\alpha, \mu_0, C_0, m, \sigma^2\}$ - which we learn using Empirical Bayes (see 5.3.1.3).

This bayesian model improves over the simple vMF mixture in multiple ways; firstly we share statistical strength by shrinking the cluster mean parameters towards a common μ_0, secondly there is flexibility to learn cluster-specific concentration parameters κ_k without the risk of overfitting if the priors are appropriately set, thirdly the posterior distribution over the parameters gives a measure of uncertainty of the parameters unlike point estimates in simple vMF mixtures. These advantages are evident in our experimental section (in section 5.7).

The likelihood of the data and the posterior of the parameter is given by,

$$P(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha, \mathbf{X}) \propto P(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi) P(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha)$$

where

$$P(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi) = \prod_{i=1}^{N} \text{Mult}(z_i|\pi) \text{vMF}(x_i|\mu_{z_i}, \kappa_{z_i})$$

$$P(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha) = P(\pi|\alpha) \prod_{k=1}^{K} \text{vMF}(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2)$$

Since the posterior distribution of the parameters cannot be calculated in closed form, we need to resort to other ways to estimate the approximate like variational inference (5.3.1) or sampling techniques (5.3.2).

5.3.1 Variational Inference

As described in section 2.4, variational inference tries to find a distribution among a class of *simple* distributions that is closest in KL divergence to the true posterior. We assume that this class of simple distributions is a product of independent distributions over the parameters. Specifically, we assume the following factored form for the approximate posterior,

$$\begin{aligned} q(\pi) &\sim \text{Dirichlet}(.|\rho) \\ q(\mu_k) &\sim \text{vMF}(.|\psi_k, \gamma_k) \quad k = 1, 2..K \\ q(z_i) &\sim \text{Categorical}(.|\lambda_i) \quad i = 1, 2..N \\ q(\kappa_k) &\equiv \text{Discussed later} \quad k = 1, 2..K \end{aligned}$$

The goal is to estimate the parameters of this factored form such that it is closest in KL divergence to the true posterior. Following standard variational inference techniques (see section 2.4 or [Wainwright and Jordan, 2008]) the parameters are estimated by solving the following optimization problem,

$$\max_{q} \qquad E_q\left[\log P(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha)\right] - E_q[\log(q)] \tag{5.1}$$

To solve the optimization problem, we use a coordinate ascent approach, where we iteratively update each parameter to maximize the objective.

$$\begin{aligned} &\log P(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}, \pi | m, \sigma^2, \mu_0, C_0, \alpha) = \log\left[\text{Dirichlet}(\pi|\alpha)\prod_{i=1}^{N}\text{Mult}(z_i|\pi)\text{vMF}(x_i|\mu_{z_i}, \kappa_{z_i})\right] \\ &= -\log B(\alpha) + \sum_{k=1}^{K}(\alpha - 1)\log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\log\pi_k + \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\left(\log C_D(\kappa_k) + \kappa_k x_i^\top \mu_k\right) \\ &+ \sum_{k=1}^{K}\left(\log C_D(C_0) + C_0\mu_k^\top \mu_0\right) + \sum_{k=1}^{K}\left(-log(\kappa_k) - \frac{1}{2}\log(2\pi\sigma^2) - \frac{(\log(\kappa_k) - m).^2}{2\sigma^2}\right) \end{aligned}$$

The expectation under q of the terms above are,

$$E_q[z_{i,k}] = \lambda_{i,k} \quad , \quad E_q[\log(\pi_k)] = \Psi(\rho_k) - \Psi\left(\sum_j \rho_j\right) \quad , \quad E_q[\mu_k] = E_q\left[\frac{\mathcal{I}_{\frac{D}{2}}(\gamma_k)}{\mathcal{I}_{\frac{D}{2}-1}(\gamma_k)}\right]\mu_k$$

where $\Psi(.)$ is the digamma function.

Optimizing 5.1 w.r.t ρ,

$$\rho_k = \alpha + \sum_{i=1}^{N} \lambda_{ik} \tag{5.2}$$

Optimizing 5.1 w.r.t $\boldsymbol{\lambda}$

$$\lambda_{ik} \propto exp(E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] + E_q[\log(\pi_k)])$$

where

$$E_q[\log \text{vMF}(x_i|\mu_k, \kappa_k)] = E_q[\log C_D(\kappa_k)] + E_q[\kappa_k] x_i^\top E_q[\mu_k] \tag{5.3}$$

Optimizing 5.1 w.r.t posterior parameters of $\boldsymbol{\mu}$

$$\mu_k \propto \exp\left(E_q[\kappa_k]\left(\sum_{i=1}^{N} E_q[z_{ik}]x_i\right) + C_0\mu_0\right)$$

$$\Rightarrow \quad \gamma_k = \left\| E_q[\kappa_k]\left(\sum_{i=1}^{N} E_q[z_{ik}]x_i\right) + C_0\mu_0 \right\|$$

$$\psi_k = \frac{E_q[\kappa_k]\left(\sum_{i=1}^{N} E_q[z_{ik}]x_i\right) + C_0\mu_0}{\gamma_k}$$

where

$$E_q[z_{ik}] = \lambda_{ik}$$

Estimating the posterior of the concentration parameters is not straightforward because of the presence of the log-bessel function in $\log C_D(\kappa)$. The difficulty in estimation arises because we need to define $q(\kappa_k)$ such that $E_{q(\kappa_k)}[\log C_D(\kappa_k)]$ (eq (5.3)) can be calculated either in closed form or numerically. However, we are not aware of any distribution for which this is possible. Therefore, we present two ways of estimating the posterior $q(\kappa_k)$ which rely on approximating $E_q[\log C_D(\kappa_k)]$ in two ways - (a) Sampling scheme (b) Bounding scheme.

5.3.1.1 Sampling scheme

For the sampling scheme we assume no specific distribution for $q(\kappa_k)$,

$$q(\kappa_k) \equiv \text{No-specific form} \quad k = 1, 2..K$$

We estimate the posterior of κ_k's (and related quantities such as $\log C_D(\kappa_k)$) by drawing samples of κ_k instead of maintaining a specific distribution. To draw samples we need the conditional distribution for κ_k. In typical MCMC or Gibbs sampling this is not a problem since the conditional distribution for κ_k is expressed in terms of samples of the other parameters. However, the

variational inference in the previous step does not maintain samples but instead maintains only posterior distributions. Therefore we need to express the conditional distribution of κ_k not in terms of samples but in terms of the posterior distribution of the other parameters. We achieve this by introducing a series of approximations to the true conditional distribution of κ_k.

The true conditional distribution of κ_k is given by,

$$\begin{aligned} P(\kappa_k|\mathbf{X}, m, \sigma^2, \mu_0, C_0, \alpha) &= \frac{P(\kappa_k, \mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha)}{P(\mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha)} \\ &\propto P(\kappa_k, \mathbf{X}|m, \sigma^2, \mu_0, C_0, \alpha) \\ &\propto \int_{\mathbf{Z}} \int_{\pi} \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\kappa}^{-k}} P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \pi|m, \sigma^2, \mu_0, C_0, \alpha) \\ &\approx E_q \left[P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \pi|m, \sigma^2, \mu_0, C_0, \alpha) \right] \end{aligned} \tag{5.4}$$

Using Jensen inequality, we have,

$$E_q \left[P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] \geq \exp\left(E_q \left[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] \right) \tag{5.5}$$

where the inner expectation (after discarding terms which do not depend on κ_k) is

$$\begin{aligned} &E_q \left[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}|m, \sigma^2, \mu_0, C_0, \pi) \right] = \\ &\left[\sum_{i=1}^{N} \lambda_{ik} \left(\log C_D(\kappa_k) + \kappa_k x_i^\top E_q[\mu_k] \right) \right] - \log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \text{ constants ..} \\ &= n_k \log C_D(\kappa_k) + \kappa_k \left(\sum_{i=1}^{N} \lambda_{ik} x_i^\top E_q[\mu_k] \right) - \log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \text{ constants ..} \end{aligned} \tag{5.6}$$

where

$$n_k = \sum_{i=1}^{N} \lambda_{ik}$$

Combining (5.4), (5.5), (5.6) the approximate conditional distribution of κ_k is

$$P(\kappa_k|\mathbf{X}, m, \sigma^2, \mu_0, C_0, \pi) \propto \exp\left(n_k \log C_D(\kappa_k) + \kappa_k \left(\sum_{i=1}^{N} \lambda_{ik} x_i^\top E_q[\mu_k] \right) \right) \text{logNormal}(\kappa_k|m, \sigma^2) \tag{5.7}$$

Having identified the proportionality of the conditional distribution of κ_k in terms of the distribution of the other parameters, we can use MCMC sampling with a suitable proposal distribution to draw samples. Based on the above expression, it is easy to see that computation

becomes easier if we use the logNormal$(.|m, \sigma^2)$ as the proposal distribution for MCMC sampling i.e. the logNormal likelihood terms cancel. However, we still recommend using a different proposal distribution such as logNormal or Normal around the *current value* of the parameter for convergence reasons (note that the κ's are relative large positive values, therefore using a Normal Proposal distribution works fine). This is because the posterior of κ_k's are reasonably concentrated around a small region and since the prior distribution is reasonably vague, simply using logNormal$(.|m, \sigma^2)$ as the proposal distribution increases the number of iterations for convergence.

The MCMC sampling step for the posterior of κ introduces flexibility into the model as the samples are now from the true posterior (given the rest of the parameters). The downside is that the variational bound is no longer guaranteed to be a valid lower-bound since $E_q[\log C_D(\kappa_k)]$ and $E_q[\kappa_k]$ are estimated through the samples. However, we observed that the posterior distribution of the κ_k's was reasonably concentrated around the mode and the estimates from samples gave good empirical performance in terms of predictive power.

This partial MCMC sampling does not increase computational costs much since there are only K variables to be estimated and the major computational bottleneck is in updating $\boldsymbol{\lambda}$. Another alternative to repeated sampling is to use grid search, where we break down the continuous posterior distribution of κ_k across a finite set of points, and use this discrete distribution to estimate the expectation of various quantities.

Note that the same model with an unnormalized prior for κ was used in [Bangert et al., 2010]. However since κ is not a natural parameter of the vMF distribution, it is not clear whether the unnormalized prior results in a valid posterior distribution.

5.3.1.2 Bounding Scheme

The core problem in doing full variational inference for the model is the computation of the $E_q[\log C_D(\kappa_k)]$ in (5.1), more specifically $E_q[\log I_{.5D-1}(\kappa_k)]$. In the bounding scheme, we upper-bound the log bessel function using a Turan type inequality [Baricz et al., 2011] followed by an approximation using the delta method. More specifically, the growth of the modified bessel function of first kind with order v and argument u i.e. $\mathcal{I}_v(u)$ can be lower-bounded by [Baricz et al., 2011],

$$u\frac{I_v(u)'}{I_v(u)} \leq \sqrt{u^2+v^2} \quad \Rightarrow \quad \frac{I_v(u)'}{I_v(u)} \leq \sqrt{1+\frac{v^2}{u}}$$

Integrating over $u > 0$,

$$log(I_v(u)) \leq \sqrt{u^2+v^2} - v\log\left(v\sqrt{v^2+u^2}+v^2\right) - v\log(u)$$

$$E_q[log(I_v(u))] \leq E_q[\sqrt{u^2+v^2}] - vE_q[\log\left(v\sqrt{v^2+u^2}+v^2\right)] - vE_q[\log(u)] \tag{5.8}$$

We assume that posterior distribution for the cluster concentration parameters (κ_k's) is log-normally distributed,

$$q(\kappa_k) \equiv \text{logNormal}(.|a_k, b_k)$$

In order to find the parameters of the posterior of the $k'th$ cluster, i.e. a_k, b_k, we optimize (5.1),

$$n_k E_q[\log C_D(\kappa_k)] + E_q[\kappa_k]\sum_i \lambda_{ik} x_i^\top E_q[\mu_k] - E_q[\log(\kappa_k)] - E_q\left[\frac{(log(\kappa_k)-m)^2}{2\sigma^2}\right] - H(q)$$

where

$$\begin{aligned}
&E_q[\kappa_k] = \exp(a_k + .5 * b_k^2)\\
&E_q[\log(\kappa_k)] = a_k\\
&E_q\left[\frac{(log(\kappa_k)-m)^2}{2\sigma^2}\right] = \frac{1}{2\sigma^2}\left(E_q[\log(\kappa_k)^2] - 2mE_q[\log(\kappa_k)] + m^2\right)\\
&E_q[\log(\kappa_k)^2] = a_k^2 + b_k^2\\
&H(q(\kappa_k)) = .5 + .5\log(2\pi b_k^2) + a_k\\
&E_q[\log C_D(\kappa_k)] = (.5D-1)E_q[\log(\kappa_k)] - .5D\log(2\pi) - E_q[\log I_{.5D-1}(\kappa_k)]
\end{aligned} \tag{5.9}$$

To calculate $E_q[\log I_{.5D-1}(\kappa_k)]$ we first apply the bound in eq (5.8) followed by the application of delta approxmiation method (note that all the expressions on the RHS of eq (5.8) are twice differentiable). The expectation of a function g over a distribution q is given by

$$E_q[g(x)] \approx g(E_q[x]) + g''(E_q[x])\frac{Var_q[x]}{2} \tag{5.10}$$

Combining (5.9), (5.10) and (5.1), we can estimate the posterior parameters a_k, b_k using gradient descent . Although it is tempting to directly apply the delta method to calculate $E_q[\log \mathcal{I}_{.5D-1}(\kappa_k)]$, this leads to expressions that are not computable directly as well as numerically unstable.

5.3.1.3 Empirical Bayes estimate for prior parameters

In many cases the end user might not have a good idea on how to set the prior parameters $\{m, \sigma, \mu_0, C_0\}$ of the model. In such situations we can use the parametric Empirical Bayes (EB) estimation technique [Casella] for estimating the prior parameters from the data itself. Using the variational lower bound as a surrogate to the marginal likelihood of the parameters, the EB

estimate for m, σ is obtained by maximizing the VLB,

$$max_{m,\sigma^2} \quad -\frac{K}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{K}(E_q[\log(\kappa_k)^2] - 2mE_q[\log(\kappa_k)] + m^2)$$

$$\Rightarrow \quad m = \frac{1}{K}\sum E_q[\log(\kappa_k)], \quad \sigma^2 = \frac{1}{K}\sum_{k=1}^{K} E_q[\log(\kappa_k)^2] - m^2$$

The empirical bayes estimate for μ_0, C_0 is given by,

$$\max_{\mu_0, C_0} \; K\log C_D(C_0) + C_0\mu_0^\top \left(\sum_{k=1}^{K} E_q[\mu_k]\right)$$

Following [Sra, 2012], the updates are given by,

$$\mu_0 = \frac{\sum_{k=1}^{K} E_q[\mu_k]}{\left\|\sum_{k=1}^{K} E_q[\mu_k]\right\|}, \quad C_0 = \frac{\bar{r}D - \bar{r}^3}{1-\bar{r}^2} \quad \text{where} \quad \bar{r} = \frac{\left\|\sum_{k=1}^{K} E_q[\mu_k]\right\|}{K}$$

The empirical bayes estimate for α is given by,

$$\max_{\alpha>0} -\log B(\alpha) + (\alpha-1)s$$

$$\text{where } s = \sum_{k=1}^{K} E_q[\log \pi_k]$$

$$B(\alpha) = \frac{\Gamma(\alpha)^K}{\Gamma(K\alpha)}$$

Since there is no closed-form solution, we need to apply a numerical optimization like gradient descent or newton's method to find the estimate for α.

5.3.2 Collapsed Gibbs Sampling

Sampling techniques, unlike variational inference estimate the posterior by repeatedly drawing samples from the conditional distribution of the parameters. The main advantage of sampling methods over variational inference is that they are guaranteed to converge to the true posterior if *sufficient* number of samples are drawn, however, the downside is that it is often difficult to judge whether sufficiency has been achieved or not.

For our models, we can develop efficient collapsed Gibbs sampling techniques for the model by using the fact that vMF distributions are conjugate w.r.t each other. This enables us to

completely integrate out $\{\mu_k\}_{k=1}^K$ and π and update the model by maintaining only the cluster assignment variables $\{z_i\}_{i=1}^N$ and the concentration parameters $\{\kappa_k\}_{k=1}^K$. The conditional distributions is as follows,

$$P(z_i = t|\mathbf{Z}^{-i}, \mathbf{X}, \boldsymbol{\kappa}, m, \sigma^2, \mu_0, C_0) =$$

$$\int_\pi \int_{\boldsymbol{\mu}} P(x_i|z_i, \mu_k, \kappa_k) P(z_i|\pi) P(\pi|\alpha) \prod_{j\neq i} P(z_j|\pi) P(x_j|\mu_{z_j}, \kappa_{z_j}) \prod_{k=1}^K P(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2)$$

$$\propto \int_\pi P(\pi|\alpha) P(z_i|\pi) \prod_{j\neq i} P(z_j|\pi) \; . \; \prod_{k=1}^K \int_{\mu_k} \left(P(x_i|z_i, \mu_k, \kappa_k) \prod_{j\neq i, z_j=k} P(x_j|\mu_k, \kappa_k) \right) P(\mu_k|\mu_0, C_0)$$

$$\propto \frac{\Gamma(K\alpha) \prod_{k=1}^K \Gamma(\alpha + n_{k,-i} + I(t=k))}{\Gamma(K\alpha + N) \prod_{k=1}^K \Gamma(\alpha)} \prod_{k=1}^K \frac{C_D(\kappa_k)^{n_{k,-i}+I(k=t)} C_D(C_0)}{C_D\left(\left\| \kappa_k \left(I(t=k) x_i + \sum_{j\neq i, z_j=k} x_j \right) + C_0\mu_0 \right\| \right)}$$

$$\propto (\alpha + n_{t,-i}) \, C_D(\kappa_t) \frac{C_D\left(\left\| \kappa_t \sum_{j\neq i, z_j=t} x_j + C_0\mu_0 \right\| \right)}{C_D\left(\left\| \kappa_t \left(\sum_{j:z_j=t} x_j \right) + C_0\mu_0 \right\| \right)}$$

Similarly, we derive the conditional distribution for κ_k,

$$P(\kappa_k|..) \propto \int_{\mu_k} \prod_{z_i=k} P(x_i|\mu_k, \kappa_k) P(\mu_k|\mu_0, C_0) \text{logNormal}(\kappa_k|m, \sigma^2)$$

$$\propto \frac{C_D(\kappa_k)^{n_k} C_D(C_0)}{C_D\left(\left\| \kappa_k \sum_{j:z_j=k} x_j + C_0\mu_0 \right\| \right)} \text{logNormal}(\kappa_k|m, \sigma^2)$$

The conditional distribution of κ_k is again not of a standard form and as before, we use a step of MCMC sampling (with log-normal proposal distribution around the current iterate) to sample κ_k.

The empirical bayes estimation in the context of MCMC sampling can be done as described in [Casella, 2001]. The prior parameters are estimated by maximizing the sum of the marginal likelihood of all the samples.

5.4 Hierarchical Bayesian von Mises-Fisher Mixtures

We assume that the user wants to organize the data into a given fixed hierarchy of nodes $\mathcal{N}$. The hierarchy is defined by the parent function $pa(x) : \mathcal{N} \rightarrow \mathcal{N}$ which denotes the parent of the node x in the hierarchy. The generative model for H-vMFmix is given by,

$$
\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_k \sim \text{vMF}(.|\mu_{pa(k)}, \kappa_{pa(k)}) \; k \in \mathcal{N} \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \; k \in \mathcal{N} \\
&z_i \sim \text{Categorical}(.|\pi), z_i \in \{\text{Leaf nodes of hierarchy }\} \\
&x_i \sim \text{vMF}(.|\mu_{z_i}, \kappa_{z_i}) \quad i = 1, 2..N
\end{aligned}
$$

The user specified parameters are $\{\mu_0, C_0\}$ - parameters at the root-node and $\{m, \sigma^2, \alpha\}$. Each node n is equipped with a vMF distribution with parameters μ_n, κ_n. The mean parameters of the sibling nodes are drawn from a common prior defined by their parent vMF distribution. The concentration parameters for all the nodes are commonly drawn from a log normal distribution with mean m and variance σ^2. The instance x_i is drawn from the one of the leaf-nodes z_i (Note that the data **X** resides on the leaf-nodes). By drawing the parameters of siblings from a common parent node, we are forcing the nodes which are closer to each other in the hierarchy to have similar model parameters. Our hope is that this would enable data to be organized into the appropriate levels of granularity as defined by the hierarchy. For inference, we can develop similar variational inference methods with Empirical Bayes step to estimate the prior parameters. The updates for all the parameters are similar except for the posterior distribution of cluster mean parameter μ_k,

For leaf nodes

$$
\begin{aligned}
\gamma_k &= \left\| E_q[\kappa_k] \left(\sum_{i=1}^{N} E_q[z_{ik}] x_i \right) + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}] \right\| \\
\psi_k &= \frac{E_q[\kappa_k] \left(\sum_{i=1}^{N} E_q[z_{ik}] x_i \right) + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}]}{\gamma_k}
\end{aligned}
$$

For non-leaf nodes

$$
\begin{aligned}
\gamma_k &= \left\| E_q[\kappa_k] \sum_{c:pa(c)=k}^{N} E_q[\mu_c] + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}] \right\| \\
\psi_k &= \frac{E_q[\kappa_k] \sum_{c:pa(c)=k}^{N} E_q[\mu_c] + E_q[\kappa_{pa(k)}] E_q[\mu_{pa(k)}]}{\gamma_k}
\end{aligned}
$$

5.5 Temporal Bayesian von Mises-Fisher Mixtures

We present Temporal vMF mixture(T-vMFmix), a state-space model based on the vMF distribution where the parameters of a cluster at a given time point have evolved from the previous time point. Given data across T time-points, $\mathbf{X} = \{\{x_{t,i}\}_{i=1}^{N_t}\}_{t=1}^{T}$ and a fixed number of clusters K, the generative model is given by,

$$
\begin{aligned}
&\pi \sim \text{Dirichlet}(.|\alpha) \\
&\mu_{1,k} \sim \text{vMF}(.|\mu_{-1}, C_0)\ k = 1, 2, 3..K \\
&\mu_{t,k} \sim \text{vMF}(.|\mu_{t-1,k}, C_0)\ \ t = 2...T,\ k = 1, 2, 3..K \\
&\kappa_k \sim \text{logNormal}(.|m, \sigma^2) \quad k = 1, 2..K \\
&z_{t,i} \sim \text{Mult}(.|\pi) \quad t = 1, 2...T; i = 1, 2, ..N_t, \\
&x_{t,i} \sim \text{vMF}(.|\mu_{z_{t,i}}, \kappa_{z_{t,i}}) \quad t = 1, 2..T;\ i = 1, 2, ...N_t
\end{aligned}
$$

The prior parameters in this model are $\{\mu_{-1}, C_0, \alpha, m, \sigma^2\}$. The cluster specific concentration parameters κ_k's are commonly drawn from a log-Normal distribution with mean m and variance σ^2. The mean parameters of the clusters at time t are drawn from a vMF distribution centered around the previous time $t-1$ with a concentration C_0. This time-evolution of the cluster parameters introduces a layer of flexibility and enables T-vMFmix to accommodate changes in the mean parameter within a given cluster. The C_0 parameter controls the sharpness of time-evolution; having a large value of C_0 ensures that cluster parameters are more or less the same over time whereas a low value of C_0 enables the cluster parameter to fluctuate wildly between adjacent time points.

$$
\begin{aligned}
&\eta_t \sim \mathcal{N}(.|\eta_{t-1}, \Sigma^{-1}), \qquad \alpha_{t,k} = \exp(\eta_{t,k}) / \sum_{j=1}^{K} \exp(\eta_{t,j}) \\
&\pi_t \sim \text{Dirichlet}(.|\alpha_t)
\end{aligned}
$$

For inference, we can develop a similar variational methods and Empirical Bayes step for estimating the prior parameters. The updates for the mean parameters at time t and cluster k are

$$
\begin{aligned}
\gamma_{t,k} &= \left\| E_q[\kappa_k] \sum_{i=1}^{N_t} E_q[z_{t,ik}] x_i + C_0 E_q[\mu_{t-1,k}] + I(t < T) C_0 E_q[\mu_{t+1,k}] \right\| \\
\psi_{t,k} &= \frac{E_q[\kappa_k] \sum_{i=1}^{N_t} E_q[z_{t,i,k}] x_i + C_0 E_q[\mu_{t-1,k}] + I(t < T) C_0 E_q[\mu_{t+1,k}]}{\gamma_{t,k}}
\end{aligned}
$$

Table 5.1: Dataset Statistics

Dataset	#Instances	#Training	#Testing	#True Clusters	#Features
TDT4	622	311	311	34	8895
TDT5	6366	3183	3183	126	20733
CNAE	1079	539	540	9	1079
K9	2340	1170	1170	20	21839

For the partial MCMC sampling for κ_k, the approximate likelihood in (5.6) is

$$E_q[\log P(\kappa_k, \mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\kappa}^{-k}, \boldsymbol{\eta}|..)] =$$
$$-\log(\kappa_k) - \frac{(\log(\kappa_k) - m)^2}{2\sigma^2} + \left(\sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,ik}]\right)\log C_d(\kappa_k) + \sum_{t=1}^{T}\sum_{i=1}^{N_t} E_q[z_{t,ik}]x_i^\top E_q[\mu_k] + constants$$

The Empirical Bayes estimate of C_0 can be computed as follows,

$$\begin{aligned}
\max_{C_0} \quad & KT\log C_D(C_0) + C_0\sum_{t=1}^{T}\sum_{k=1}^{K} E_q[\mu_{t,k}]^\top E_q[\mu_{t-1,k}] \\
\Rightarrow s &= \frac{\sum_{t=1}^{T}\sum_{k=1}^{K} E_q[\mu_{t,k}]^\top E_q[\mu_{t-1,k}]}{KT} \\
\bar{r} &= ||s|| \\
C_0 &= \frac{\bar{r}D - \bar{r}^3}{1-\bar{r}^2}
\end{aligned}$$

Note that C_0 determines the level of influence of the previous time-step to the next time-step. In some sense it acts as a regularization term forcing the parameters of the next time-step to be similar to the previous one. We recommend that this be set manually than being learnt from the data. The reason is that since the estimation is done through maximizing the marginal likelihood, C_0 typically tends to 0, i.e. towards no regularization.

5.6 Experimental Setting

5.6.1 Datasets

Throughout our experiments we used several popular benchmark datasets (Table 5.1),

1. **TDT-{4,5}** [Allan et al., 1998] : A corpus of reuters news articles over different spans of time. The news articles are labeled to denote different events that transpired during that time. We used only those subset of documents for which labels were available.

2. **CNAE**[1]: A collection of 1080 small-text documents of business descriptions of Brazilian companies categorized into a subset of 9 categories like medical, legal, industrial etc.

3. **K9** [2]: A collection of webpages that appeared on Yahoo! news portal over several days under 20 different topics.

All datasets (except NIPS) have associated class-labels and are single-labeled i.e. each instance is assigned to exactly one class-label. For all the datasets, we performed standard stopword removal and stemming, followed by Tf-Idf weighting and normalization to a unit-sphere.

5.6.2 Evaluation Metrics

Analysing the effectiveness of clustering algorithms is a difficult problem. Although clustering tries to achieve high intra-cluster similarity and low inter-cluster similarity, this in itself might not lead to good clusters with respect to the end application. Due to the inherent subjectivity, conclusive comparisons between clustering algorithms often involve user studies which are time consuming and not always be feasible. Despite our reservations, we will attempt to evaluate the clustering algorithms using two commonly used types of metrics,

1. **Recoverability**: In scenarios where we have a fixed set of instances and we are confident that the clustering algorithm has the right modelling assumptions and can ideally recover the ground-truth clusters, *recoverability* tells us how well the predicted clusters match with the ground-truth clusters.

   We choose two popularly used recoverability metrics - Normalized Mutual Information and Adjusted Rand Index (more details in Appendix B).

2. **Generalizability** (or *Predictive power*): Generalizability measures how well does the clustering algorithms generalize to new unseen data. We used likelihood on a held-out test-set to compare clustering algorithms. A better clustering algorithm is the one that is able to explain the data better and assigns a higher likelihood to unseen data.

Both the measures have been widely used (although previous work have rarely reported both). It is possible that *generalizability* and *recoverability* might in itself be two conflicting goals i.e. a clustering method that has high recoverability need not generalize well and vice versa.

[1] http://archive.ics.uci.edu/ml/datasets/CNAE-9
[2] http://www-users.cs.umn.edu/boley/ftp/PDDPdata/

The validity of evaluating models like H-vMFmix or D-vMFmix is particularly questionable as their goal is to provide some insightful view of the data and not necessarily generalize to new data or recover existing clusters. However, following previous works [Blei et al., 2004],[Blei and Lafferty, 2007] we present the predictive power of these methods on held out test-set for the sake of completeness.

5.6.3 Description of Baselines

We compared our proposed models against the following baselines,

1. **vMFmix**: A mixture of vMF distributions described in Section 5.2. Note that this is similar to the model developed in [Banerjee et al., 2006], except that all clusters share the same κ. Using the same κ for all clusters performed significantly better than allowing cluster-specific κ_k's - this may be due to the absence of Bayesian prior.

2. **K-means (KM)**: The standard k-means with euclidean distance and hard-assignments.

3. **Gaussian Mixtures (GM)**: A mixture of Gaussian distributions with the means commonly drawn from a Normal prior and a single variance parameter for all clusters - using cluster specific variances performed worse (even with an Inverse gamma prior).

4. **Multinomial Mixture (MM)**: A graphical model where each instance is represented as a vector of feature counts, drawn from a mixture of K Multinomial distributions.

5. **Latent Dirichlet Allocation (LDA)** [Blei et al., 2003] An admixture model where the words in an instance are drawn from an instance-specific mixture of K multinomials. Note that LDA may not be a good choice to ensure a single label for an instance, however we still present the results for the sake of completeness.

We used the Tf-Idf normalized data representation for vMFmix, KM and GM, and feature counts representation (without normalization) for MM and LDA. For all the methods, every instance is assigned to the cluster with largest probability before evaluation. Also, for the sake of a fair comparison, we started all the clustering algorithms with the same *random* initial cluster-assignment variables $\mathbf{Z}$. All the results presented in this chapter are averaged over 10 different runs. Note that the kmeans++ initialization helped in increasing robustness to local maxima (in most cases, all the 10 runs lead to very similar clustering objectives).

5.6.4 Convergence Criterea and Other Details

We used the following convergence criteria for the inference procedures,

1. The increase in objective was less than .1.
2. The difference in norm between adjacent iterations was less than .01 successively for 3 iterations.
3. The number of iterations reached 200.

For sampling the concentration parameters, we used a log-normal proposal distribution with mean as the current iterate and variance 1.0. The first 300 iterations were used for burn in and the last 200 iterations was used to estimate the samples (500 iterations seemed to be good enough). Since there are only K concentration parameters this does not affect the computational complexity at all.

Since all the algorithms are EM based algorithms and the objective is not convex, it is necessary to initialize the variables appropriately. For all the vMF mix models we initialized the concentration parameters with relative low values like 5 or 10 which implies that the initial spread of the clusters are very broad. Similarly the variance parameter of the Gaussian mixture was set to a very high value to make the initial distribution very broad. This performed much better than setting the initial value of the variance parameters to the MLE of all the data.

Naively calculating $I_\nu(a)$ is not possible because of the presence of ratio of factorial terms in the bessel function, which causes numerical overflow issues for even relatively small $\nu \sim 15$. Therefore, we used the log approximation given in [Hornik and Grün] and exponentiated it,

$$\begin{aligned}\log I_\nu(a) &\approx \sqrt{a^2+(\nu+1)^2} + (\nu+\frac{1}{2})\log\frac{a}{\nu+\frac{1}{2}+\sqrt{a^2+(\nu+1)^2}} - \frac{1}{2}\log(\frac{a}{2}) \\ &\quad + (\nu+\frac{1}{2})\log\frac{2\nu+\frac{3}{2}}{2(\nu+1)} - \frac{1}{2}\log(2\pi) \\ I_\nu(a) &= \exp(\log I_n u(a))\end{aligned}$$

5.7 Results

We first present a full fledged comparison between the various clustering methods based on recoverability, followed a within vMF family comparison based on generalizability metrics. We finally present some more experimental analysis such as the effect of normalization and the ability to detect out-of-sample data of the clustering methods.

5.7.1 Recoverability

First and the most natural question that we would like to answer is 'are vMF mixtures any better than standard Gaussian or Multinomial mixtures?'. Table 5.2 shows the results for B-

Table 5.2: Recoverability of the various clustering algorithms on five datasets using 20 clusters. Each result is averaged over 10 different runs of the EM algorithm.

Dataset	Metric	B-vMFmix	vMFmix	Kmeans	GM	MM	LDA
CNAE	*NMI*	**0.748**	0.650	0.605	0.620	0.726	0.320
	ARI	**0.669**	0.426	0.357	0.311	0.528	0.169
K9	*NMI*	**0.551**	0.543	0.482	0.487	0.487	0.269
	ARI	**0.352**	0.350	0.293	0.264	0.321	0.110
TDT4	*NMI*	**0.900**	0.880	0.857	0.842	0.720	0.841
	ARI	**0.799**	0.729	0.693	0.675	0.467	0.692
TDT5	*NMI*	**0.860**	0.851	0.833	0.827	0.796	0.776
	ARI	**0.710**	0.676	0.617	0.591	0.580	0.501
CITESEER	*NMI*	0.401	**0.403**	0.402	0.408	0.271	0.033
	ARI	**0.141**	**0.141**	0.136	0.134	0.052	0.000

vMFmix, vMFmix, KM, GM, MM and LDA on five datasets data sets with 20 clusters using two recoverability metrics - NMI and ARI. The full set of results with varying clusters and metrics can be found in [Gopal and Yang, 2014b]. Note that no separate test set is needed for this type of evaluation, i.e., all the data (but the labels) are used for training the models and the labels are used for evaluating the resulting clusters. For all the methods, we used the MAP estimate of the parameters to ensure a fair comparison.

Both the vMF based models, vMF-mix and B-vMFmix perform better than the the non-vMF clustering algorithms on almost all datasets and metrics. The conclusions and differences in performance are similar even with varying number of clusters. Despite the reservations that we might have about the clustering metrics, a superior performance on most datasets certainly provides evidence of superiority of vMF based methods.

5.7.2 Generalizability

We present the results of likelihood based evaluation only within the vMF family. We do not compare outside the vMF family since the support of the models are different (vMF models are defined on unit spheres, Multinomial models are defined on non-negative integers, etc) and the resulting likelihoods are not numerically comparable.

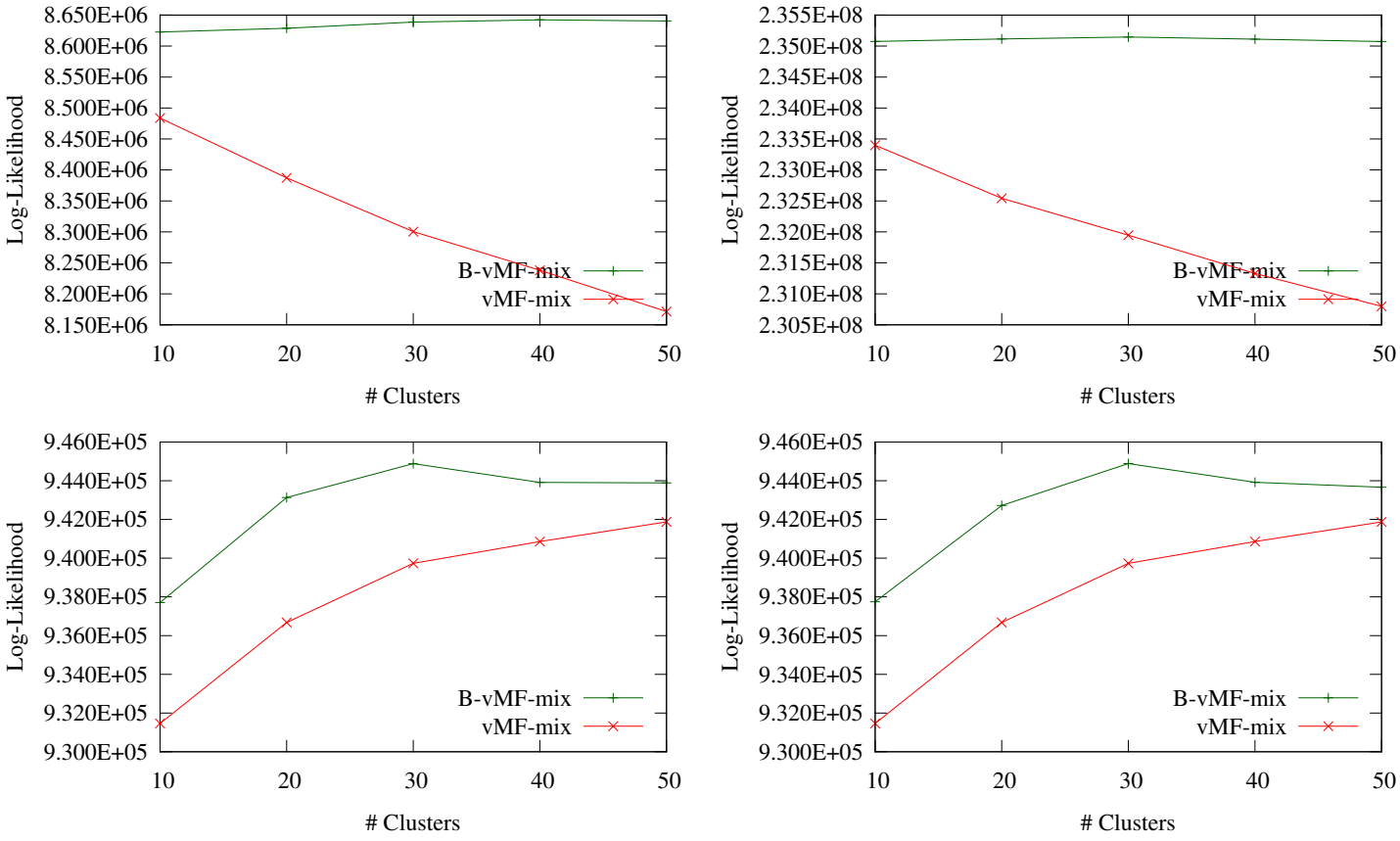


Figure 5.1: The Likelihood on the held out test-set (averaged over 10 runs) assigned by B-vMFmix and vMFmix for varying number of clusters on 4 datasets - TDT4 (top left), TDT5 (top right), K9 (bottom left), CNAE (bottom right).

5.7.2.1 Comparison of B-vMFmix and vMFmix

We compare the performance of the basic vMF mixture (vMFmix) and the fully Bayesian vMF mixture model (B-vMFmix) on four data sets by varying the number of clusters from 10 to 30 respectively. The better model will be able to assign higher likelihood for the test data. The train-test splits are shown in Table 5.1. The B-vMFmix is trained through variational inference (using *Sampling method* for the concentration parameters) and vMFmix is trained through the EM algorithm. In our experiments, the *sampling method* performed better than the *bounding method* with an insignificant difference in running time, we therefore report all results using the sampling based method. All the prior parameters are learned from the data using Empirical Bayes. The results are averaged over 10 runs with different initial values. The results (Figure 5.1) show that B-vMFmix is able to assign a higher likelihood for unseen data on all 4 datasets.

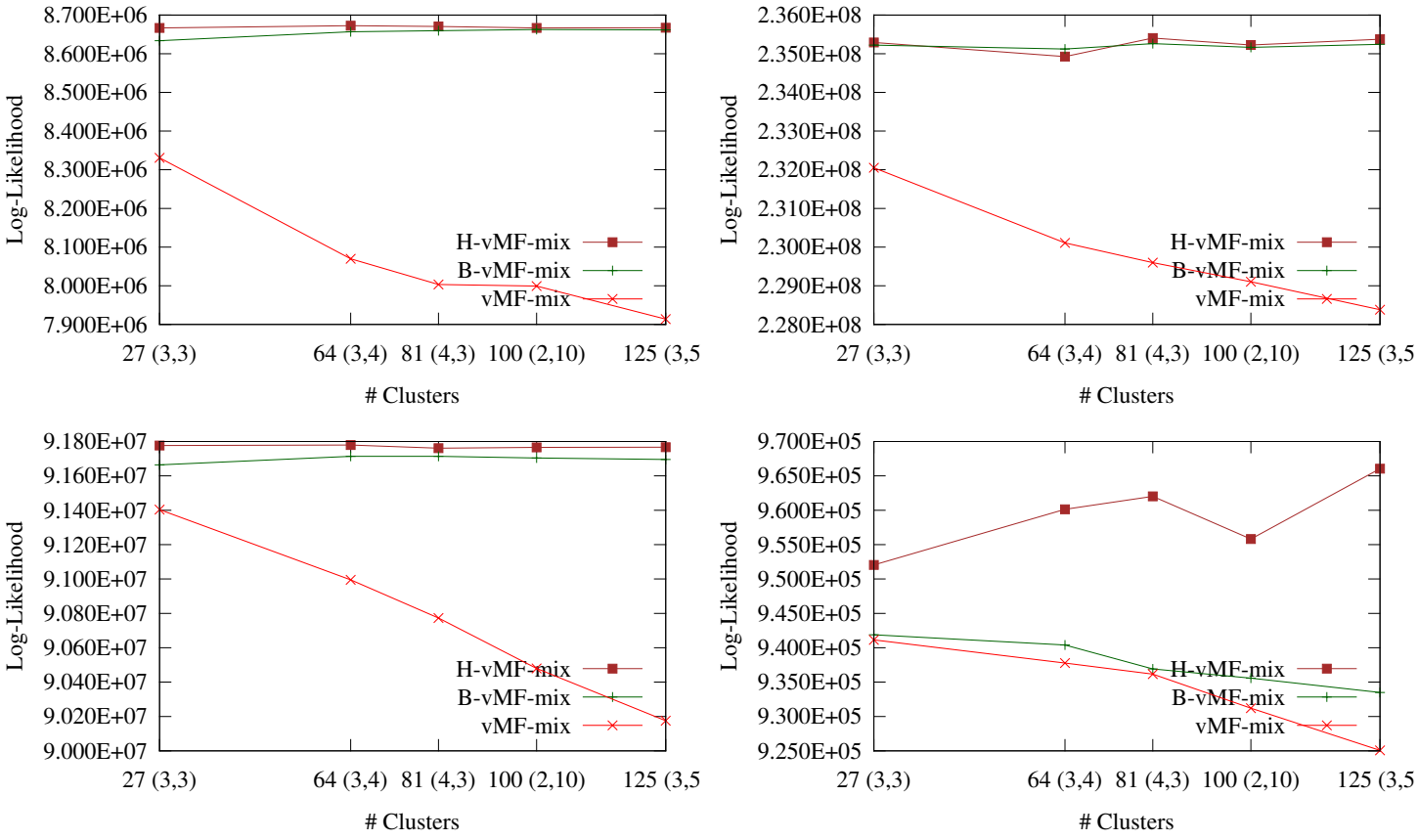


Figure 5.2: Average Likelihood of H-vMFmix, B-vMFmix and vMFmix for different input hierarchies. The number of clusters for B-vMFmix, vMFmix is set to number of leaf nodes in the hierarchy - TDT4 (top left), TDT5 (top right), K9 (bottom left), CNAE (bottom right).

5.7.2.2 Comparison of H-vMFmix against B-vMFmix and vMFmix

Figure 5.2 compares the performance of the hierarchical vMF mixture model (H-vMFmix) with vMFmix and B-vMFmix. We test H-vMFmix on 5 different hierarchies with varying depth of hierarchy and branch factor (branch factor is the #children under each node). For example (height h=3,branching factor b=4) is a hierarchy 3 levels deep where the zeroth level is the root-node under which there are 4 children which forms the first level, each of these 4 children in turn have 4 children each leading to 16 nodes in the second level, each of these 16 children have 4 children each which forms 64 children at the third level. We present on the results on (h=3,b=3),(h=3,b=4),(h=4,b=3),(h=2,b=10),(h=3,b=5) for H-vMFmix. We also plot the performance of the corresponding vMFmix and B-vMFmix; the number of clusters was set equal to the number of leaf nodes in the hierarchy. The results show that H-vMFmix is able to offer better predictive power than vMFmix and B-vMFmix.

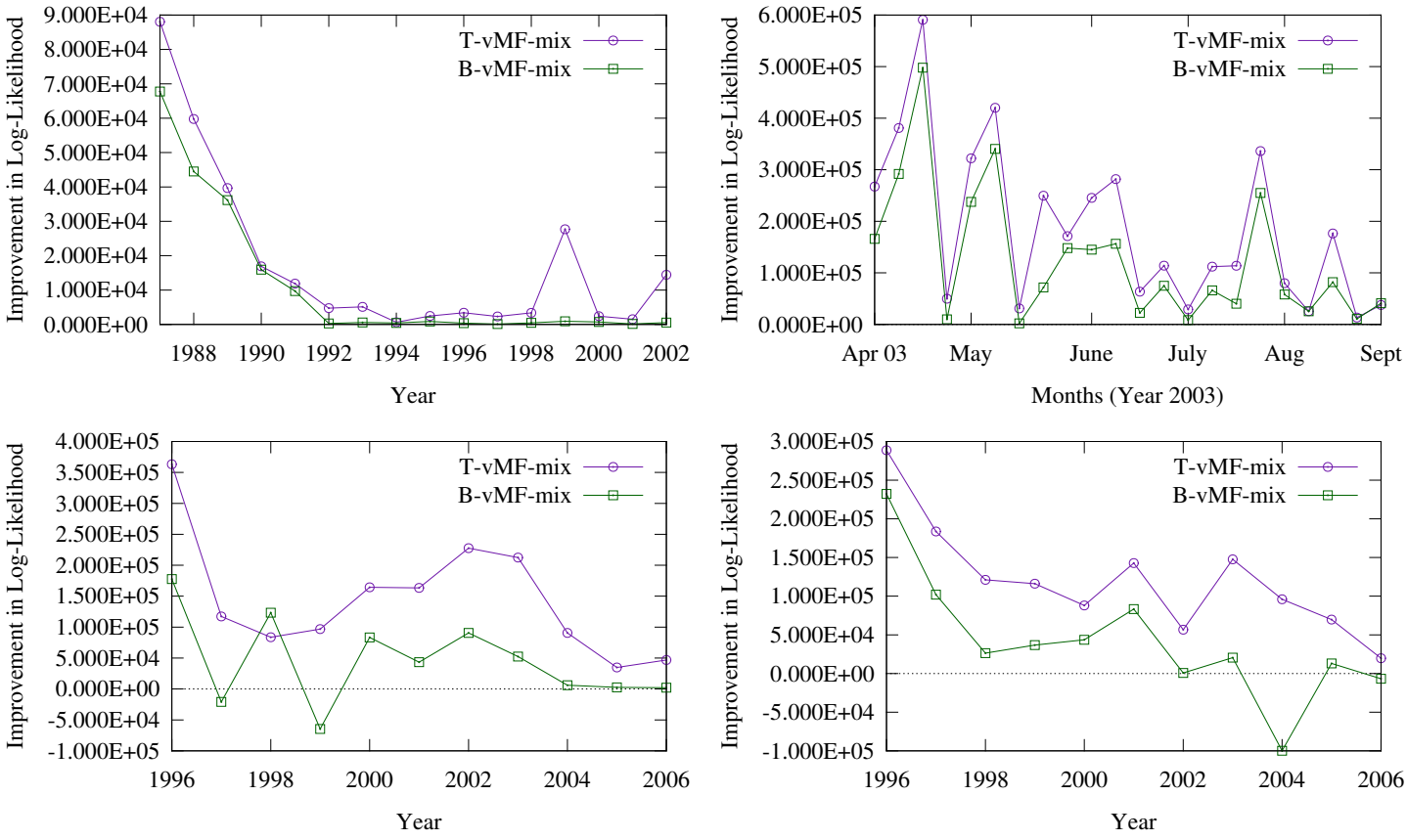


Figure 5.3: Relative improvement in likelihood of T-vMFmix, B-vMFmix models over vMFmix across time (with 30 clusters) - NIPS dataset (top left), TDT5 (top right), NYTC-politics (bottom left) and NYTC-elections (bottom right).

5.7.2.3 Comparison of T-vMFmix against B-vMFmix and vMFmix

We compare the predictive power of T-vMFmix against B-vMFmix and the baseline vMFmix on the following four temporal datasets which have time-stamps associated with instance,

1. **NIPS** : This dataset has 2483 research papers from the proceedings of the NIPS conference spread over a span of 17 years. The collection was divided into 17 time-points based on year of publication of each paper.

2. **TDT5** : This dataset has 6636 news stories over a span of few months (Apr 03 to Sept 03). The collection was divided into 22 time-points where each time-point corresponds to one week of news. Note that we selected only those news stories which have been judged by humans to belong to one or more preselected news 'events' [Allan et al., 1998].

Table 5.3: Comparison of different data representations using vMFmix and K-means using 30 clusters. Bold face numbers indicate best performing data representation.

Method	Dataset	CNAE		TDT5	
	Representation/Metric	NMI	ARI	NMI	ARI
vMFmix	Tf-Idf normalization	**.650**	**.426**	**.851**	**.676**
	Tf normalization	.553	.310	.827	.657
K-means	Tf-Idf normalization	**.605**	**.311**	**.827**	**.591**
	Tf normalization	.536	.288	.799	.551

3. **NYTC-{politics,elections}**: This is a collection of New York Times news articles from the period 1996-2007. We formed two different datasets - the first one being more general than the second one. The first subset consist of all news articles which have been tagged as politics and the second subset consists of articles which have been tagged elections. For computational reasons, we removed all words which occurred less than 50 times and news stories which have less than 50 words. This leads to the nytc-politics dataset consisting of 48509 news articles with 16089 words and the nytc-elections dataset consisting of 23665 news articles and 10473 words.

We test the methods by predicting the likelihood of the data of the next time-point given all the articles from the beginning using 30 topics. For ease of visualization, we plot the relative improvement in log-likelihood of the B-vMFmix and T-vMFmix models over the simple vMFmix model (This is because the log-likelihood from adjacent time-points are not comparable and fluctuate wildly). The results (Figure 5.3) suggests that T-vMFmix by taking the temporal nature into account, is able to always assign a higher likelihood to the next time-point than B-vMFmix and vMFmix.

5.7.3 Analysis of Tf-Idf Weighting

In order to fully understand the benefits of inverse document frequency scaling, we compared the performance of representing the data using plain Tf normalization against Tf-Idf normalization with 'ltc' [3] based term weighting. The results of experiments on TDT5 and CNAE datasets using vMFmix and K-means (as reported in Table 5.3) show that on both the datasets representing the data using Tf-Idf normalization gives significant performance benefits.

[3] http://nlp.stanford.edu/IR-book/html/htmledition/document-and-query-weighting-schemes-1.html

Table 5.4: Accuracy of in-sample vs out-of-sample prediction of the various models.

Method	Accuracy
BvMFmix	**0.612**
vMFmix	0.558
GM	0.582
MM	0.552
LDA	0.601

5.7.4 In-sample vs Out-sample Detection

The likelihood based comparisons in the previous section can be essentially viewed as a density estimation task where the goal of the clustering algorithm is to estimate a density function that can assign probabilities to unseen data. One of the side-effects of this model is that it can be used to distinguish between in-sample and out-of-sample data. One way to do this is as follows - given a new instance, we compute the likelihood of this instance and check if it is inside or outside the 95% confidence interval of the average likelihood of in-sample instances; if it is inside, we consider it as an in-sample instance, otherwise it is considered an out-of-sample instance.

In this subsection, we particularly analyze the ability of the clustering models to make such distinctions. We used 5,529 research articles from the Arxiv *computer science* field to build a density estimation model and tested the ability of the model to correctly predict the membership of a test set of 11,131 research articles from both Arxiv *computer science* and *quantitative biology* fields. We trained five models - BvMFmix, vMFmix, GM, MM and LDA using 20 clusters. For all the models other than MM and LDA, we used Tf-Idf normalization followed by dimension reduction using SVD to 50 dimensions and unit normalization. For MM and LDA, we removed all words that occurred less than 20 times. Note that we performed dimension reduction/feature selection step because it improved the results of detection (not because of computational reasons). The results of the predictions, tabulated in Table 5.4, show that B-vMFmix offers the best detection accuracy followed by LDA.

5.7.5 Sampling vs Bounding Concentration parameters

In this subsection we compare the two ways of estimating the posterior distribution of κ's int the variational inference algorithm - the sampling based method and the bounding based method. In theory, the sampling based method should be more accurate in estimating the posterior distribution but taking a longer time than the bounding based method. However, in practice

Table 5.5: Sampling vs Bounding method for estimating posterior - Likelihood and computational time taken on the K9 dataset using 20 clusters (averaged over 10 runs).

Method	Likelihood	Time (in secs)
Sampling	91683824.75	25.44
Bounding	91509341.59	24.58

(Table 5.5) we found that the sampling based method gives better likelihood in approximately the same time as the bounding based method. This is because bulk of the computation time of the variational algorithm is spent in computing the updates for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Although just between sampling and bounding, the former *is* slower than latter; the benefit of using bounding is masked by time taken in performing the other updates.

5.8 Summary

In this chapter we developed a suite of Bayesian vMF models that are better alternatives to the existing approaches based on multinomial or Gaussian distributions. We developed a flat Bayesian vMF mixture and extended to it a hierarchical and a temporal vMF mixture models. Our models are analagous to existing topic models based on multinomial distributions, but specifically suited to unit-normalized data (especially arising in text and image data).

The fast variational/sampling algorithms make the methods scalable to reasonably large data volumes with high-dimensional feature spaces. The experiments provide strong empirical support for the effectiveness of our approaches - all our models outperformed strong baselines (k-means, Multinomial Mixtures and Latent Dirichlet Allocation) by a large margin on most data sets.

Transformation-based Clustering with Supervision

6

6.1 Introduction

One of the fundamental problems of unsupervised clustering algorithms such as K-means, Gaussian mixtures, von Mises-Fisher mixtures, etc the is that they do not take into account the user expectations or preferences over different views of the data.

In this chapter we address this issue by developing methods that can inject such supervision into the clustering process. The task is as follows - the user provides supervision by revealing the cluster assignments for some subset of ground-truth clusters, the goal is to design a system which uses this supervision to understand user expectations i.e. the view the user is interested in, and discover *similar* unobserved clusters in the unlabeled data. The key idea in our solution is to learn a transformation of the data such that the clusters in the transformed data space match the given supervision (i.e. user expectations/preferences). This learnt transformation is then applied on the unlabeled data thereby folding them into space where the unobserved clusters are consistent with user expectations. We propose two novel ways of learning the transformation -

1. **Conditional Probability Maximization** (CPM) We learn a transformation of data under a generative model for the instances. The generative process used is the standard probabilistic clustering assumption that each instance is drawn from one of the latent clusters with the specified probability density, for e.g. Gaussian. The transformation function G is learnt from the supervision in a **discriminative manner** by maximizing the conditional probability of the cluster label given the transformed instance.

   The framework is generally applicable to any probabilistic clustering model as long as a suitable a suitable G can be defined. In this chapter we will focus on its particular appli-

cation to Gaussian mixtures (GM) and von Mises-Fisher mixtures (VM). In the GM case, we define G to be a linear transformation matrix, and for VM G is a linear transformation followed by projection to unit-sphere. For both models, we develop techniques to estimate G and related parameters.

2. **Cluster Separability Criterion** (CSC) In this framework we do not make any parametric assumptions about the class conditional densities of the instances. We setup a constrained optimization problem to estimate a linear transformation of the data such that every pair of clusters are *adequately* separated. Specifically, we introduce constraints that every pair of cluster means be separated by a margin proportional to the radius of the cluster. The hope is that the estimated linear transformation would preserve similar properties for the unobserved clusters and making them easily separable.

In both cases, we use the given supervision (in the form of labeled instances instances and true cluster labels for some subset of ground-truth clusters) to estimate the transformation. We then apply the learned transformation function to the unlabeled data for clustering in the next step. Since our proposed methods rely on using a transformation function as bridge to reshape the discovery of unobserved clusters, we call our framework Transformation-based Clustering with Supervision - TCS framework.

We conducted thorough evaluations and tested our framework on 23 data sets from different application domains such as time-series, text, handwriting recognition, face recognition, etc. We have observed substantial and consistent improvement in performance (in five clustering metrics) over other competing methods.

6.2 Related Work

There are two primary areas of related work that use supervision to improve clustering of unlabeled data - Probabilistic Constrained Clustering (PCC) and Distance Metric Learning (DML).

PCC methods [Wagstaff et al., 2001, Basu et al., 2002] try to inject supervision using a probabilistic generative model for the instances. The instances $\mathbf{X}$ consists of both the labeled part $\mathbf{X}_L$,the unlabeled part $\mathbf{X}_U$ and the cluster assignments $\mathbf{Z} = \{\mathbf{Z}_U \cup \mathbf{Z}_L\}$. The observed cluster assignments $\mathbf{Z}_L \subset \mathbf{Z}$ is used as supervision, i.e., the constraints. The model parameters $\boldsymbol{\theta}$ and the latent cluster assignments $\mathbf{Z}_U$ are estimated by maximizing the likelihood of the data $\mathbf{X}$ subject to the fixed $\mathbf{Z}_L$,

$$\max_{\boldsymbol{\theta}, \mathbf{Z}_U} \quad \log P(\mathbf{X}|\mathbf{Z}_L, \mathbf{Z}_U, \boldsymbol{\theta}) \tag{6.1}$$

However, when the task is to detect previously unobserved cluster, this optimization reduces to plain clustering. This can be seen by rewriting the objective as a sum of the labeled objective (which is constant) and the unlabeled objective (which is just standard clustering) as:

$$\max_{\boldsymbol{\theta}_L} \log P(\mathbf{X}_L|\mathbf{Z}_L, \boldsymbol{\theta}_L) + \max_{\boldsymbol{\theta}_U, \mathbf{Z}_U} \log P(\mathbf{X}_U|\mathbf{Z}_U, \boldsymbol{\theta}_U)$$

Clearly, there is no *learning* nor transfer of information from the observed clusters to the unobserved clusters. There are other works in PCC where supervision is represented in the form of pairwise constraints instead of cluster labels, i.e. the constraints are defined as whether two instances should be put in the same cluster or not. These methods optimize eq (6.1) with an additional penalty term if the constraints are not obeyed. The penalty is introduced in the form of priors [Lu and Leen, 2005], [Basu et al., 2006] or explicitly modeled using some loss function [Basu et al., 2004], [Bilenko et al., 2004]. Despite the different variations in formulating the constraints, PCC methods [Wagstaff et al., 2001, Basu et al., 2002, 2004, 2006, Lu and Leen, 2005] have the same fundamental limitation, i.e., the supervised information from the *observed* clusters is not used to reshape the discovery of *unobserved* clusters. The clustering of the unlabeled part of the data reduces to standard unsupervised clustering.

A natural extension of PCC that addresses some of its limitations is to use a more *Bayesian* approach of sharing parameters across clusters. For example, one could use a Gaussian mixture model where each cluster k has its own mean parameter θ_k, but all clusters share a common covariance matrix $\boldsymbol{\Sigma}$. Here the covariance matrix serves as the bridge to transfer information from the observed clusters to the unobserved clusters. One can envision more sophisticated models with common hyperpriors, e.g. a Gaussian hyperprior for the means and an Inverse Wishart hyperprior for the covariances. To our knowledge, such Bayesian revisions of PCC have not been studied in the context of discovering new clusters in unlabeled data (which is the focus of this chapter). As we will show in our experiments (where we implemented such a Bayesian PCC model as a baseline), this way of sharing information is not as effective as directly fitting for the cluster labels in the transformed space, which we propose in the TCS framework.

In DML [Blitzer et al., 2005, Xing et al., 2002, Goldberger et al., 2004] the objective is to learn a distance metric that respects the supervision which is provided in the form pairwise constraints. More specifically, given a set of labeled clusters, the distance metric is estimated such that it pulls the within-cluster pairs towards each other and pushes the cross-cluster pairs away from each other. DML methods differ from each other in how the loss functions are defined over the pairwise constraints such as the hinge loss [Blitzer et al., 2005], Euclidean-distance loss [Xing et al., 2002], log loss [Goldberger et al., 2004], etc.

DML has been typically used in the context of nearest-neighbor classification but not in the context of discovering *unobserved* clusters. We argue that existing DML methods have two problems w.r.t to discovering unobserved clusters: Firstly, DML optimizes for pairwise distances and is therefore 'unaware' of the clustering criterion (the objective function for clustering) used. This can lead to overfitting, for example, even if the data is optimally clustered, DML would still try to increase inter-cluster distances and decrease intra-cluster distances. Secondly, optimizing for different loss functions (e.g., hinge loss or Euclidean loss) do not necessarily yield a metric that is also optimal for clustering. Explicitly fitting for the cluster-labels (or maximizing the margin between clusters) without resorting to surrogate measures like pairwise constraints between instances is the fundamental difference between our TCS models and other DML methods.

Indirectly related to our work are a few works in discriminative clustering [Krause et al., 2010], [Xu et al., 2004] and constrained spectral clustering [Rangapuram and Hein, 2012], [Wang and Davidson, 2010], [Lu and Carreira-Perpinán, 2008]. These former methods use a discriminative objective like that of SVM for clustering, but do not handle supervision. The latter methods incorporate supervision in a spectral clustering framework, but cannot scale beyond a few thousands of instances. For a thorough discussion of the drawbacks of spectral clustering framework see [Nadler and Galun, 2007] and reference therein. It is also worth mentioning that some other work like [Joulin et al., 2010], [Finley and Joachims, 2005] reformulate the classification problem as a clustering one, but however cannot discover previously unobserved clusters in data.

6.3 The Need for Supervision

Any typical clustering task involves making atleast two assumptions - number of clusters (or the prior parameters if using Bayesian non-parametric methods) and the distance measure, both of which determine the type of clusters generated. In a probabilistic model, the distance measure is determined by the choice of the distribution for e.g. Euclidean distance corresponds to Gaussian with Identity covariance matrix, cosine-similarity corresponds to von Mises-Fisher distribution etc. Typically, the user's subjective choice of the distance measure (or probability distribution) does not match the ground-truth and the generated clusters do not match user expectations.

To demonstrate this, we ran two popular clustering algorithms - the Gaussian Mixture model (GM) and the von Mises-Fisher mixture model (VM), which use different probability distributions and optimize different likelihoods, on the 20newsgroups dataset [1] with 20 clusters using

[1] qwone.com/ jason/20Newsgroups/

Table 6.1: Likelihood at Local optimum reached by EM vs Likelihood at ground-truth (Local optimum is better!)

Algorithm →	GM	VM
Local optimum	**-18115**	**4.09965e+09**
Ground-truth	-18168	4.09906e+09

the EM algorithm. We compared the likelihoods (the clustering objective) of the algorithms under two settings,

1. The likelihood at a local optimum reached by EM [2].
2. The likelihood when the true labels are given, i.e. cluster assignments fixed to the ground-truth.

As table 6.1 shows, the likelihood obtained at a local optimum is better than ground-truth - the ground-truth is suboptimal ! The clustering algorithm is optimizing for something else other than ground-truth. This means that the user expected clusters can never be recovered by these clustering algorithms. This is a clear case of mismatch between what the user expects and the user specified distance measure. It is precisely this mismatch that we hope to reduce by using some partial supervision from the user.

6.4 Learning Transformations by CPM

We are provided supervised training data from K clusters, $\mathcal{S} = \{x_i, t_i\}_{i=1}^{N}$ where $x_i \in \mathcal{X}$, $t_i \in \{1, 2...K\}$ and unlabeled examples $\mathcal{U}$. For convenience define $y_{ik} = I(t_i = k)$. Given a probabilistic generative model using a mixture of K distributions $\{f(x|C_k)\}_{k=1}^{K}$ where C_k denotes the parameters of cluster K, CPM estimates the transformation function G as follows,

$$\arg\max_{G} \log P(\mathbf{Y}|\mathbf{C}^{mle}(G), G(\mathbf{X}))$$

$$\text{where} \quad \mathbf{C}^{mle}(G) = \arg\max_{\mathbf{C}} P(G(\mathbf{X})|\mathbf{C}, \mathbf{Y}),$$

$$P(\mathbf{Y}|G(\mathbf{X}), \mathbf{C}^{mle}(G)) = \prod_{i=1}^{N}\prod_{k=1}^{K}\left[\frac{f(G(x_i)|C_k^{mle}(G))}{\sum_{k'=1}^{K} f(G(x_i)|C_{k'}^{mle}(G))}\right]^{y_{ik}}$$

[2]The EM algorithm was intialized with the ground-truth cluster assignments

Here $\mathbf{C}^{mle}$ denotes the maximum likelihood estimates of the cluster parameters in the transformed space i.e the cluster parameters that best explain the transformed data $G(\mathbf{X})$. G on the other hand is estimated by maximizing the conditional likelihood of the cluster-labels given $\mathbf{C}^{mle}$ i.e. the transformation that best explains the cluster-labels. Together this ensures that we learn G such that the optimal cluster parameters $\mathbf{C}^{mle}$ also optimally fit the cluster-labels $\mathbf{Y}$. The transformation G is then applied to $\mathcal{U}$ thereby folding it into a space where hopefully the clusters match user expectations. Note it would be easy to incorporate uncertainty or confidence scores in the supervision by appropriately weighting the probability of each instance to the corresponding cluster.

Unlike typical clustering algorithms, our TCS framework has a *learning* component as well i.e. we learn how to discover unobserved clusters from supervision. Since any learning algorithm has chances of overfitting, we add an additional regularization term. Together,

$$\begin{aligned} [\text{OPT1}] \quad & \max_{G,\mathbf{C}} \quad \log P(\mathbf{Y}|\mathbf{C}(G), G(\mathbf{X})) - \gamma\lambda(G) \\ & \text{s.t} \quad \frac{\partial \log P(G(\mathbf{X})|\mathbf{C}, \mathbf{Y})}{\partial \mathbf{C}} = 0 \end{aligned}$$

where γ is the regularization parameter and λ is the regularization function. Note that the second constraint is another way to say $\mathbf{C}$ is the MLE estimate of $G(\mathbf{X})$. In the following subsections, we discuss how to estimate G with two choices for $f(x|C_k)$ - Gaussian and von Mises-Fisher. In Appendix C, we also discuss the estimation procedure for Gamma distributions as well as outline the general procedure for any mixture distribution.

6.4.1 Gaussian Mixtures (TCS-GM)

We define a simple mixture of K Gaussians with unit variance and cluster means $\{\theta_k\}_{k=1}^K$ over $\mathcal{R}^P$ where

$$f(x|\theta_k) = \frac{1}{\sqrt{2\pi}} \exp(-\|x - \theta_k\|^2)$$

The transformation function is defined as $G(x) = Lx$, a linear transformation using matrix $L \in \mathcal{R}^{P\times P}$. The parameters $\{\theta_k\}_{k=1}^K$ and transformation L is estimated by solving OPT1.

$$[\text{OPT1}] \quad \max_{L,\boldsymbol{\theta}} \quad \log P(\mathbf{Y}|\boldsymbol{\theta}, L\mathbf{X}) - \gamma\lambda(L) \tag{6.2}$$

$$\text{s.t} \quad -\frac{1}{2}\frac{\partial \sum_{i=1}^{N} y_{ik}\|Lx_i - \theta_k\|^2}{\partial \theta_k} = 0 \tag{6.3}$$

where,

$$P(y_{ik}=1|\theta_k, Lx_i) = \frac{e^{-\frac{1}{2}(Lx_i-\theta_k)^\top(Lx_i-\theta_k)}}{\sum_{k'=1}^{K} e^{-\frac{1}{2}(Lx_i-L\theta_{k'})^\top(Lx_i-L\theta_{k'})}}$$

$$\text{and} \quad \log P(\mathbf{Y}|\boldsymbol{\theta}, L\mathbf{X}) = -\frac{1}{2}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} y_{ik}\|Lx_i-\theta_k\|^2\right] - \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\exp(-\frac{1}{2}\|Lx_i-\theta_k\|^2)\right)$$

If m_k denotes the mean of all instances which belong to cluster k, the equality constraint in OPT1 is simply $\theta_k = Lm_k$. Substituting this into the optimization problem and rewriting as a minimization problem,

$$\min_{L} \quad \gamma\lambda(L) + \frac{1}{2}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} y_{ik}\|Lx_i-Lm_k\|^2\right] + \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\exp(-\frac{1}{2}\|Lx_i-Lm_k\|^2)\right)$$

This is a nonconvex function in L. However, it can be rewritten as a convex optimization problem in terms of $A = L^\top L$ with a positive definite constraint on A. Assuming the regularizer is convex, this leads to a convex semidefinite program,

$$\begin{aligned}\min_{A} \quad & F(A) = \gamma\lambda(A) + \frac{1}{2}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} y_{ik}d_A(x_i,m_k)\right] + \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\exp(-\frac{1}{2}d_A(x_i,m_k)\right)\\ \text{s.t} \quad & A \succeq 0\end{aligned}$$

where $d_A(a,b) = (a-b)^\top A(a-b)$.

For the regularizer $\lambda(A)$, we tried different choices

1. $\|A-I\|^2$ (Frobenius Norm from Identity)
2. $\|A\|_2^2$ (Frobenius Norm from zero)
3. $\|A\|_*$ (Nuclear Norm)
4. $trace(A) - log(det(A))$ (Log-det divergence)
5. $\|A-I\|_1$ (Entrywise-L1 Norm)

Different regularizers favor different kinds of linear transformations, for e.g., Nuclear norm [Ma et al., 2011] favours lower rank solutions, Entrywise-L1 norm favours sparser transformations, the log-det regularlizer also prefers lower rank solutions but penalizes sum of log of

Algorithm 3 Accelerated gradient descent for TCS-GM.

Input: $\{\mathbf{X}, \mathbf{Y}\}$, step-length sequence S_t
Initialize: Define $H_t = I, \beta_t = 1$
while not converged **do**
    $A_t = \mathcal{PSD}(H_t - s_t \nabla F(H_t))$
    $\beta_{t+1} = \frac{1+\sqrt{1+4\beta_t^2}}{2}$
    $H_{t+1} = A_t + \frac{\beta_t - 1}{\beta_t}(A_t - A_{t-1})$
end while
Output: A_t
$\mathcal{PSD}$ denotes projection to the positive semidefinite cone

eigenvalues instead [Davis et al., 2007] etc. The choice of the regularizer ultimately depends on the data and kind of clusters the user seeks.

For differentiable regularizers, the optimization can be solved using projected gradient descent where we take a step along the direction of the negative gradient (with the stepsize determined by backtracking line search) and then project the update back into the positive semidefinite cone. We observed that we could significantly improve the speed by using accelerated gradient descent [Beck and Teboulle, 2009] instead (Algorithm 3). For the non-differentiable regularizers we can use projected subgradient instead. These algorithms provably converge to the optimal solution if the step size is appropriately set [Boyd and Vandenberghe, 2004]. The gradient of the objective can be succintly written as,

$$\nabla F(A) = \gamma \lambda'(A) + \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - p_{ik}(A)) d_{ik} d_{ik}^{\top}$$
$$\text{where } p_{ik}(A) = P(y_{ik} | L x_i, \theta_k)$$

We found that our customized parallel solver using accelerated projected gradient descent worked much faster than existing SDP solvers. The solution in A recovers L upto any rotation matrix. This is because $A = L^{\top} L$ can always be rewritten using $A = L^{\top}(Q^{\top} Q)L$, for any rotation matrix $Q^{-1} = Q^{\top}$. However rotating the data corresponds to changing the basis and does not affect the clustering algorithm.

Note that A is very different from the seemingly similar covariance matrix of a Gaussian distribution. Firstly, unlike the covariance matrix, A is not parameter of the distribution and does not make the distribution sum to 1. Secondly, covariance matrices are typically estimated by maximizing $P(\mathbf{X}|\mathbf{Y})$, this includes supervised versions such as linear discriminant analysis (LDA) and Fisher LDA [Hastie et al., 2009]. The transformation matrix A on the other hand, is

optimized to fit the labels $P(\mathbf{Y}|L\mathbf{X}, \theta)$ and therefore unlike LDA, A does not have a closed form solution.

6.4.2 von Mises-Fisher Mixtures (TCS-VM)

The density function of a von Mises-Fisher distribution is defined over $\mathcal{X} \equiv \{x : \|x\| = 1, x \in \mathcal{R}^P\}$ and is given by

$$f(x|\mu, \kappa) = \frac{\kappa^{(.5P-1)}}{(2\pi)^{.5P}\mathcal{I}_{.5P-1}(\kappa)} \exp(\kappa \mu^\top x)$$

where $\mathcal{I}_\nu(a)$ is the modified bessel function of first kind with order ν and argument a.

As in the Gaussian mixtures case, we consider a mixture of K vMF distributions with mean parameters $\{\mu_k\}_{k=1}^K$ and unit concentration parameter where

$$f(x|\mu_k) = \frac{\exp(\mu_k^\top x)}{(2\pi)^{.5P}\mathcal{I}_{.5P-1}(1)}$$

Since the support of vMF is only over the unit-sphere, any transformation function should ensure the transformed space still lies on the unit-sphere. Therefore we define the transformation function G as a linear transformation with matrix $L \in \mathcal{R}^{P\times P}$ with normalization,

$$G(x) = \frac{Lx}{\|Lx\|}$$

With this transformation function, the optimization problem OPT1 is a non-convex function in L. Note that the cluster mean parameters $\{\mu_k\}_{k=1}^K$ have a closed form expression for the MLE estimate in terms of L and $\mathbf{X}$. However unlike the GM case, we do not recommend substituting it into the objective of OPT1 as it leads to computationally intensive expressions. We instead resort to an alternating optimization between μ_k's and L as shown in Algorithm 4. The optimization step in line 5 is nonconvex in L and can solved using gradient descent to converge to a locally optimal solution. We found that in practice it gave good empirical results.

6.5 Learning Transformations by CSC

In the CSC framework we do not make any parametric assumptions about the conditional distribution of the cluster-labels. The transformation function $G(x) = Lx$ is a linear transformation matrix which is estimated directly by enforcing cluster separability conditions.

More specifically, we formulate an optimization problem where every pair of cluster means is constrained to be separated by margin proportional to the sum of the trace of covariance

Algorithm 4 Optimizing $L, \{\mu_k\}_{k=1}^{K}$ for TCS-VM.

1: **Input:** $\{\mathbf{X}, \mathbf{Y}\}, T$ iterations
2: **Initialize:** L
3: **for** $t = 1, ..T$ **do**
4: **update** $\mu_k = \frac{\sum_{i=1}^{N} y_{ik} n_i}{\| \sum_{i=1}^{N} y_{ik} n_i \|}$ where $n_i = \frac{Lx_i}{\|Lx_i\|}$
5: **update** $L = \arg\min_L$
$$\gamma\lambda(L) + \sum_{i=1}^{N}\sum_{k}^{K} y_{ik}\left[\frac{x_i^\top L^\top \mu_k}{\|Lx_i\|} - \log\left(\sum_{j=1}^{K}\exp(\frac{x_i^\top L^\top \mu_j}{\|L^\top x_i\|})\right)\right]$$
6: **end for**

matrices of the respective clusters. The optimization problem is parameterized in terms of $A = L^\top L$ as follows,

$$\begin{aligned} \min \quad & \|A - I\|^2 \\ s.t \quad & (m_i - m_j)^\top A (m_i - m_j) \geq \gamma(trace(AC_i) + trace(AC_j)) \quad \forall i, j = 1..K, i \neq j \\ & A \succeq 0 \end{aligned}$$

Here m_k and C_k denote the mean and covariance of instances assigned to cluster k. The trace of a matrix is the sum of its eigenvalues; $trace(AC_i)$ intuitively captures the total spread of the cluster along its basis eigenvectors. The constraints mean that the distance between the cluster means should be separated by a distance proportional to the sum of volumes of the two clusters. The γ parameter is a user-specified proportionality constant to control the degree to which the separability constraints must be enforced; for e.g. $\gamma = 0$ would mean all constraints are satisfied by default and the optimal solution is $A^* = I$. Similar to TCS-GM one can also use different regularizers instead of $\|A - I\|^2$ such as log-det divergence etc.

To solve this optimization problem one can use interior point methods which has become the defacto method for solving constrained SDPs. In this we work, we used the SeDuMi[3] package which solves the equality constrained dual problem using the barrier method. Note that our separability constraints is partially inspired by some of the assumptions that previous work [Dasgupta, 1999],[Kannan et al., 2005] have made for providing some provable guarantees on recovering clusters.

[3] sedumi.ie.lehigh.edu/

6.6 Experimental Setting

For all the experiments, we consider the class-labels associated with the dataset to be the user expected ground-truth clusters. We randomly partitioned the classes into three sets - training, validation and testing. The methods TCS-GM, TCS-GM-L2, TCS-CS and LMNN use the validation set for tuning the regularization parameter.

We assume the number of clusters in the unlabeled data is known, if not, well established techniques like AIC, BIC or nonparametric versions of clustering using dirichlet process priors can be used. All the results are averaged over 50 different restarts with Kmeans++ initialization [Arthur and Vassilvitskii, 2007].

As in any matrix learning method, for high dimensional datasets, learning (or even storing) a full matrix L is computationally intensive. Previous literature have identified three ways to tackle this issue,

1. Learn a diagonal L instead of full matrix L. This drastically improves the scalability, but at the cost of flexibility in the set of transformations [Weinberger and Saul, 2009].

2. Directly learn a low rank transformation, i.e. $L \in \mathcal{R}^{r \times P}$ where $r \ll P$ instead of a full rank matrix. However, the optimization problem now becomes nonconvex in terms of this low rank L (see [Journée et al., 2010]).

3. Reduce the dimension of the data using Singular Value Decomposition (SVD), followed by learning a full rank matrix on the low dimensional data.

In our experiments, we found that solution (3) worked best. Infact dimension reduction using SVD sometimes **improved** the clustering performance on multiple datasets since it removes the intrinsic noise in the data. This is agreement with several observations in practice [Zha et al., 2001], [Drineas et al., 2004] and in theory [Ding and He, 2004],[Kannan et al., 2004]. Therefore, on datasets with more than 200 dimensions, we used SVD to fold the data into a 30 dimensional space (refer Section 6.8.4 for detailed experiments on how SVD affects clustering).

6.6.1 Description of Baselines

1. **GM**: A mixture of K Gaussian distributions which share a single common variance parameter. All parameters are estimated using EM algorithm. Note that this model is unsupervised and cannot use supervision.

2. **TCS-GM** : Our proposed conditional probability maximization technique with Gaussian mixture model as discussed in section 6.4.1. The regularizer was $\|A - I\|^2$.

3. **TCS-CS** : Our proposed cluster separability criterion where the linear transformation is estimated using a constrained optimization framework as discussed in section 6.5.

4. **TCS-GM-L2** : Our proposed TCS technique using Gaussian mixture model as discussed in section 6.4.1. The regularizer was plain L2 regularization - $\|A\|^2$. Note this regularizer favors lower dimensional spaces than TCS-GM.

5. **LMNN** (Large margin Nearest Neighbor): [Weinberger and Saul, 2009] A distance metric learning technique which aims to learn a metric such that each instance's neighborhood contains only instances from the same class. If not, the closest same class neighbors are pulled and imposter classes are pushed out. The distances are penalized based on a hinge-loss. To ensure competitive performance, the number of target neighbors was set to high value - 50. This is followed by GM on the linearly transformed unlabeled data. Note that LMNN is a stronger baseline than other DML methods like Relevant Component Analysis [Shental et al., 2006], Neighborhood Component Analysis [Goldberger et al., 2004], Linear Discriminant Analysis [Fisher, 1936], etc.

6. **PCC** (Probabilistic Clustering with Constraints): [Bilenko et al., 2004] A metric learning approach which pulls within cluster elements towards each other and pushes out of cluster elements away each other. To enable sharing of information, we parameterize the metric by a single A which is estimated as follows,

$$\min_A f(A) = \sum_{i,k} y_{ik}\|x_i - \theta_k\|_A + \sum_{i,j,t_i=t_j} \|x_i - x_j\|_A - \sum_{i,j,t_i\neq t_j} \|x_i - x_j\|_A$$
$$A \succeq 0$$

   As formulated, it is a non-convex minimization with a positive-semi-definite constraint on A. Note that without the positive-definite constraint, this problem has a closed-form expression. We used the authors proposed strategy of starting from the closed-form solution and adding the smallest ϵI to make it positive-definite.

7. **BP** (Bayesian Prior Model): This is our proposed Bayesian model for TCS where there is sharing of information between the parameters. We assume the following generative process for the data,

$$\theta_k \sim \mathcal{N}(\theta_0, \sigma_0^2 I)$$
$$z_i \sim \text{Categorical}(\frac{1}{K}, \frac{1}{K}, .., \frac{1}{K})$$
$$x_i \sim \mathcal{N}(\theta_{z_i}, \Sigma)$$

Here $\theta_0, \sigma_0^2, \Sigma$ are shared between all the clusters. We assume that the hidden cluster assignments for some of the data is given as supervision. We used the familiar constrained EM algorithm [Basu et al., 2002] to derive point estimates for the unknown parameters. We also tried other variants such as cluster-specific covariance matrices with a shared Inverse Wishart hyperprior, but it did not yield any appreciable improvements.

8. **FD** [Fisher, 1936],[Fukunaga, 1990] (Fisher Discriminant analysis): We use the multiclass extension to the standard two class fisher discriminant analysis. The rows of the transformation matrix are the eigenvectors e of the following generalized eigenvalue problem,

$$\begin{aligned} S_B e &= \lambda S_W e \\ \text{where } S_B &= \sum_{k=1}^{K} n_k (m_k - m)(m_k - m)^\top \\ S_W &= \sum_{k=1}^{K} n_k C_k + \gamma I \end{aligned} \tag{6.4}$$

Here n_k, m_k and C_k refer to the number of instances, the mean and the covariance of cluster k respectively; m denotes the overall mean of all the instances. Following [Friedman, 1989] we introduce a regularization term γ to ensure that the covariance matrices have full rank. Note that rank of S_B is K, therefore only atmost K dimensions are meaningful.

6.6.2 Description of Datasets

We extensively tested our proposed methods on a wide range of datasets from several application domains such as text, time-series, speech, character recognition, face recognition etc. We provide a brief description about the datasets from each domain.

6.6.2.1 Time-series datasets

Dataset Name	#Instances	#Dimension	#Classes
Australian Signs (Aussign)	2565	352	95
Character Trajectory (Char)	2858	192	20
Daily and Sports Activity (DSPA)	152	1440	19
Libras	360	90	15

1. *Australian Signs* [4] The data consists of sample of hand movements to denote Australian Sign Language signs. There are 27 examples of each of 95 signs, each of which were captured from a signer using position trackers and instrumented gloves. There were 22 features such as x-position, y-position, etc measured over time. We used Fast Fourier Transform (FFT) to extract the first 16 important dimensions for each feature. We represented each instance using a 16x22 = 352 dimensional feature space.

2. *Character Trajectory* [5] Each instance represents the pentip trajectory (defined by the position and velocity of the pentip) over time to draw a character. We used FFT to extract the first 64 dimensions for each of the 3 directions of the pentip's velocity profile. The classes refer to the character drawn.

3. *Daily and Sports Activity Recognition* [Altun and Barshan, 2010] This datasets consists of 19 activities performed by humans in a sports hall, like walking, running etc. Each of the 19 activities is repeated multiple times by different people. Each instance represents a sequence of sensor measurements over time while the activity was performed. The goal is to cluster the sensor measurement profile according to the activity performed. We used FFT on each of the sensor and extracted a 32 dimensional vector, which was concatenated across all sensors leading to 1440 dimensional space.

4. *Libras* [6] The data set is similar to Aussign but the classes are from the brazilian sign language instead of australian. The data contains 15 signs of 24 instances each.

6.6.2.2 Text data

Dataset Name	#Instances	#Dimension	#Classes
CNAE	1079	856	9
K9	2340	21839	20
TDT4	622	8895	34
TDT5	6355	20733	125

[4] http://www.cse.unsw.edu.au/ waleed/phd/
[5] http://archive.ics.uci.edu/ml/datasets/Character+Trajectories
[6] http://archive.ics.uci.edu/ml/datasets/Libras+Movement

1. *CNAE* [7] A collection of 1080 small-text documents of business descriptions of Brazilian companies categorized into a subset of 9 categories like medical, legal, industrial etc.

2. *K9* [8] This is a collection of 2340 web pages which appeared in the Yahoo news hompage. The webpages were categorized into one of 20 broad news topics which was defined by Yahoo!.

3. *TDT-4,TDT-5* [9] These articles were collected from Reuters over a period of several months. The news stories were categorized into one of many news 'events' which happened at that time. We selected only those news articles which had relevance judgements.

Note that all the text datasets were subjected to standard stopword removal, stemming and 'ltc' term weighting [Manning et al., 2008].

6.6.2.3 Handwriting Recognition

Dataset Name	#Instances	#Dimension	#Classes
Penbased Recognition (Penbased)	10992	16	10
Letter Recognition (Letter)	20000	16	26
USPS	9298	1984	10
Binary Alpha Digits (Binalpha)	1404	2480	36
Optical Recognition (Optrec)	5620	496	10
MNIST	70000	6076	10

1. *Penbased Recognition* Each instance represents a digit from 1 to 10 drawn on a Wiacom tablet. The features are constructed by resampling the positions of the Wiacom pen and normalizing them. Refer to website [10] on how exactly the features are constructed.

2. *Letter Recognition* [11] Each instance is a black-and-white rectangular pixel display of one of the 26 capital letters in the English alphabet. The features are different statistical moments of the image such as mean, variance, correlation etc.

[7] http://archive.ics.uci.edu/ml/datasets/CNAE-9
[8] http://www-users.cs.umn.edu/boley/ftp/PDDPdata/
[9] http://www.itl.nist.gov/iad/mig/tests/tdt/
[10] http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
[11] http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

3. *USPS* [12] The dataset refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. Each instance is represented by a 16x16 grayscale image. We extracted HOG features from each pixel with a patchsize of 2x2 and concatenated to form a 1984 dimensional space.

4. *Binary Alpha Digits* [13] These are 20x16 grayscale images of 26 alphabets + 10 numbers. We extracted HOG features from each pixel with a patchsize of 2 and concatenated them.

5. *Optical Recognition* [14] Each instace represents a number from 0 to 9. The 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of 'on' pixels are counted in each block. We then used the HOG features with a patchsize of 2 leading to a 496 dimensional feature space.

6. *MNIST* [15] It is one of the most popular handwritten digits recognition datasets. Each image is a number from 0 to 9 and is represented by a 28x28 bitmap with every pixel taking a value from 0 to 255. We extracted HOG features from each bitmap with a patchsize of 2. This leads to a 6076 dimensional feature-space.

6.6.2.4 Face Recognition

Dataset Name	#Instances	#Dimension	#Classes
AT&T faces	400	19964	40
UMIST	575	10304	20
Faces96	3016	19375	151
Labeled Faces in Wild (LFW)	29791	4324	158

1. *AT & T Faces* [16] A set of face images of 40 distinct subject, with ten different images of each subject. The size of each image is 112x92 pixels with 256 grey levels per pixel. We extracted HOG features from each pixel with a patchsize of 4 and concatenated them.

2. *UMIST Faces* [17] Consists of 564 cropped grayscale images of size 112x92 of 20 people. The images covers a range of poses from profile to frontal views, and a range of subjects

[12] http://statweb.stanford.edu/ tibs/ElemStatLearn/data.html
[13] http://www.cs.nyu.edu/ roweis/data.html
[14] http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/
[15] http://www.cs.nyu.edu/ roweis/data.html
[16] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[17] http://www.cs.nyu.edu/ roweis/data.html

from different race/sex/appearance. We extracted HOG features from each pixel with a patchsize of 4 and concatenated them.

3. *Faces96 dataset* [18] This is another face recognition dataset with 20 196x196 grayscale images with different variations in the subject profile. We extracted HOG features with a patchsize of 4.

4. *Labeled Faces in Wild* [19] This is one of the largest collection of faces of popular people with 13233 images of 5749 people. We did some filtering and removed all subjects with less than 10 images, this leaves with 29791 images of 158 people. We then used HOG features to represent each image.

6.6.2.5 Speech/Speaker Recognition

Dataset Name	#Instances	#Dimension	#Classes
Isolet	7797	617	26
Wall Street Journal (WSJ)	34942	25	131

1. *Isolet* [20] This is a collection of speech recordings from 152 people who spoke the name of each alphabet twice. The classes here denote the alphabet that was spoken. The features include spectral coefficients; contour features, sonorant features, pre-sonorant features, and post-sonorant features.

2. *Wall Street Journal* [21] This is a database of oral narration by 131 people of 34942 passages from the wall street journal. Each recording is sampled and 25 MFCC's extracted and averaged over time.

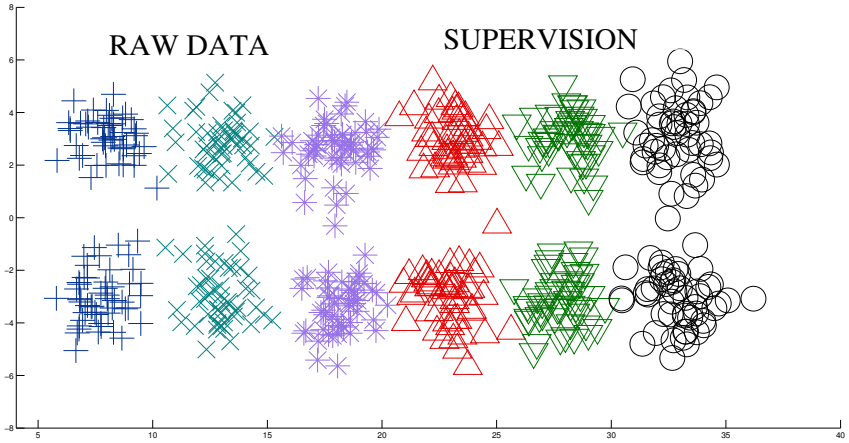
6.6.2.6 Other datasets

Note that the results of the models on these datasets are presented in the Appendix.

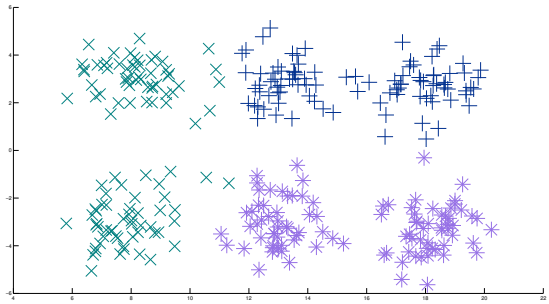
1. *Image* [22] Each instance is a 3x3 image patch denoting one of the 7 outdoor scenes in the image database. The features represent properties of the patch such region centroid, mean color, saturation etc.

[18] http://cswww.essex.ac.uk/mv/allfaces/faces96.html
[19] http://vis-www.cs.umass.edu/lfw/
[20] http://archive.ics.uci.edu/ml/datasets/ISOLET
[21] http://catalog.ldc.upenn.edu/LDC95S23
[22] http://archive.ics.uci.edu/ml/datasets/Image+Segmentation

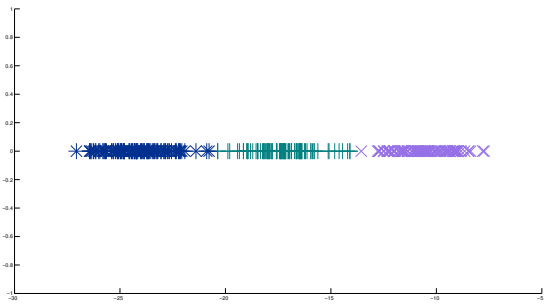
Dataset Name	#Instances	#Dimension	#Classes
Image	2310	18	7
Vowel	990	10	11
Leaves	1599	192	100



(a) Data from of a mixture of six bimodal Gaussians



(b) Clusters generated by GM on raw data. The generated clusters do not match ground-truth.



(c) Clusters generated by TCS-GM on raw data using supervision. The generated clusters match ground-truth

Figure 6.1: The benefits of using supervision when the data does not match user expectations.

2. *Vowel* [23] is popular dataset which contain speech utterances from 48 speakers. The 11 classes denote different vowel sounds.

3. *Leaves* [Mallah et al., 2013] The instances are pictures of 100 different types of leaves described by three sets of 64 dimensional vectors - shape, texture, margin, which we concatenated to form a single 192 dimensional feature space.

[23] http://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/vowel/vowel.names

Table 6.2: The performance (in terms of NMI and ARI) of the methods on the time-series datasets

Dataset	Metric	*Supervised Learning*							*Unsupervised*
		Proposed Methods			**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Aussign	*NMI*	0.890	0.910	0.925	0.931	**0.937**	0.775	0.838	0.817
	ARI	0.694	0.742	0.764	0.768	**0.809**	0.473	0.588	0.546
Char	*NMI*	0.752	**0.782**	0.754	0.668	0.671	0.690	0.737	0.687
	ARI	0.648	**0.687**	0.652	0.505	0.572	0.588	0.622	0.568
DSPA	*NMI*	0.685	**0.768**	0.734	0.739	0.597	0.668	0.642	0.678
	ARI	0.406	**0.488**	0.518	0.468	0.253	0.392	0.340	0.398
Libras	*NMI*	0.608	**0.660**	0.642	0.649	0.498	0.599	0.592	0.512
	ARI	0.474	**0.537**	0.540	0.522	0.375	0.474	0.454	0.376
Average	*NMI*	9.54%	**16.85%**	14.16%	11.74%	−0.58%	2.70%	5.04%	
Improvement	*ARI*	17.32%	30.57%	**32.12%**	21.50%	3.04%	3.68%	5.84%	

6.7 Results

6.7.1 Analysis on Synthetic Dataset

First, we show how supervision can be helpful using a small synthetic dataset. We generated data from a mixture of six clusters, where each cluster is a bimodal Gaussian distribution. The clusters lie along the x-axis, where as the modes of the Gaussians are stretched out on the Y-axis (Figure 6.1a). These are the clusters the user expects to see in the data.

Consider the task of clustering the raw data from first 3 clusters - the darkblue, cyan and purple clusters. Without any supervision, using unsupervised GM results in a clustering show in Figure 6.1b. Clearly there is confusion between clusters 2 and 3 and the user expectations are not met. On the other hand, if we use TCS-GM with clusters 4, 5, 6 as supervision, we learn a transformation that successfully captures user expectations i.e. the transformation simply collapses all points to the X-axis. When we apply this learnt transformation to the raw data and cluster using GM, we can accurately recover the ground-truth (Fig 6.1c)

6.7.2 Results on Real-world Data

We present only a part of the results in this section (using the NMI and ARI metric) and leave the complete set of results to the appendices. For ease of presentation, we tabulate the results

Table 6.3: The performance (in terms of NMI and ARI) of the methods on the Text datasets

Dataset	Metric	*Supervised Learning*							*Unsupervised*
		Proposed Methods			**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
CNAE	*NMI*	**0.608**	0.528	0.239	0.401	0.314	0.431	0.481	0.485
	ARI	**0.574**	0.405	0.202	0.384	0.194	0.285	0.355	0.359
K9	*NMI*	**0.581**	0.562	0.405	0.410	0.375	0.562	0.510	0.561
	ARI	**0.615**	0.384	0.456	0.185	0.178	0.367	0.259	0.365
TDT4	*NMI*	**0.905**	0.902	0.689	0.892	0.900	0.891	0.894	0.898
	ARI	**0.826**	0.824	0.441	0.808	0.807	0.794	0.798	0.809
TDT5	*NMI*	**0.699**	0.696	0.695	0.697	0.675	0.692	0.694	0.692
	ARI	0.198	**0.200**	0.199	0.198	0.192	0.193	0.196	0.194
Average	*NMI*	**7.68%**	2.52%	-25.34%	-11.05%	-17.66%	-2.93%	-2.52%	
Improvement	*ARI*	**33.14%**	5.74%	-15.43%	-10.10%	-24.62%	-5.61%	-7.62%	

of the models domain-wise. Overall, across all domains, we found that our proposed TCS based models achieved the best performance in 19 out of the 23 datasets.

In the time-series domain TCS-CS and TCS-GM-L2 perform the best by showing a 16.85% and 32.12% average improvement over unsupervised GM in NMI and ARI respectively. On all the text-based datasets, only TCS-GM and TCS-CS show any improvement at all. The rest are mostly negatively impacted by supervision. In the handwritten characters domain (*Penbased, Letter, USPS, Binalpha, Optrec* and *MNIST*), TCS-GM and TCS-GM-L2 achieve the best performance; TCS-GM seems to be more suited to HOG based features whereas TCS-GM-L2 works better with pixel-based statistical features. The improvement of TCS-GM is particularly higher than the rest in the handwritten characters and text datasets indicating that probably the Gaussian assumptions were suitable in these domains.

In the face clustering tasks (*AT & T, Umist, Faces96* and *LFW*) both TCS-CS and FD show good performance with 26.93% and 46.98% improvement in NMI and ARI respectively. However, we believe that images from most of the datasets except LFW are highly contrived as they were captured in ideal lighting and posing conditions. The LFW dataset, on the other hand, represents a more *realistic* distribution of images as found on the web. On this dataset, TCS-CS works best with a 36.1% improvement.

In the speech domain, on both the datasets TCS-CS achieves the best performance. The results on *WSJ* particularly highlight the importance of having supervision with a 46% improvement over unsupervised GM.

Table 6.4: The performance (in terms of NMI and ARI) of the methods on the Handwritten Character datasets.

Dataset	Metric	*Supervised Learning*							*Unsupervised*
		Proposed Methods			**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Penbased	*NMI*	0.556	0.572	**0.604**	0.464	0.571	0.401	0.487	0.522
	ARI	0.509	0.501	0.504	0.395	**0.531**	0.362	0.417	0.454
Letter	*NMI*	0.478	0.484	0.504	**0.549**	0.434	0.265	0.373	0.351
	ARI	0.301	0.301	0.329	**0.385**	0.245	0.116	0.193	0.175
USPS	*NMI*	**0.845**	0.81	0.455	0.444	0.785	0.807	0.786	0.815
	ARI	**0.829**	0.776	0.427	0.38	0.757	0.776	0.725	0.784
Binalpha	*NMI*	**0.794**	0.755	0.703	0.734	0.703	0.725	0.68	0.719
	ARI	**0.669**	0.607	0.55	0.599	0.567	0.559	0.509	0.545
Optrec	*NMI*	**0.936**	0.727	0.727	0.392	0.905	0.864	0.914	0.912
	ARI	**0.956**	0.719	0.791	0.377	0.939	0.88	0.929	0.924
MNIST	*NMI*	**0.842**	0.832	0.701	0.366	0.717	0.553	0.741	0.83
	ARI	**0.885**	0.877	0.719	0.276	0.724	0.508	0.740	0.875
Average	*NMI*	**10.15%**	5.30%	−3.82%	−18.51%	2.12%	−14.41%	−3.32%	
Improvement	*ARI*	**19.54%**	11.79%	3.70%	−10.38%	6.99%	−16.52%	−4.48%	

6.7.3 Performance of von Mises-Fisher Mixtures

Normalization to a unit sphere has often shown to work very well for text data, especially vMF-based models have been particularly effective in generating good clusters (see chapter 5) than the typical Gaussian mixtures. We present a subset of results of the vMF model outlined in section 6.4.2 on the text domain[24] in Table 6.7 On all tested datasets, TCS-VM performs significantly better than the other competing methods.

6.8 Further Experimental Analysis

6.8.1 Effect of Amount of Supervision

We analyze how the quality of clusters generated depend on the amount of supervision provided. We used a subset of the *WSJ* dataset with 25 training and 25 testing clusters for this task.

[24]Refer the supplementary material of [Gopal and Yang, 2014a] for complete set of results on Normalized data

Table 6.5: The performance (in terms of NMI and ARI) of the methods on Face clustering datasets.

Dataset	Metric	*Supervised Learning*							*Unsupervised*
		Proposed Methods			**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
AT & T	*NMI*	0.843	**0.918**	0.879	0.862	0.842	0.822	0.852	0.834
	ARI	0.627	**0.758**	0.692	0.692	0.652	0.59	0.640	0.613
Umist	*NMI*	0.588	0.806	0.739	**0.826**	0.792	0.569	0.563	0.554
	ARI	0.355	0.593	0.548	**0.676**	0.600	0.355	0.332	0.322
Faces96	*NMI*	0.922	**0.944**	0.929	0.942	0.939	0.887	0.894	0.886
	ARI	0.728	**0.789**	0.752	0.783	0.771	0.662	0.673	0.663
LFW	*NMI*	0.388	**0.415**	0.415	0.414	0.331	0.312	0.297	0.285
	ARI	0.081	**0.095**	0.089	0.089	0.040	0.030	0.026	0.021
Average	*NMI*	11.86%	**26.93%**	22.31%	26.01%	16.51%	2.71%	2.22%	
Improvement	*ARI*	7.45%	42.27%	32.17%	**46.98%**	36.33%	2.12%	3.01%	

Table 6.6: The performance (in terms of NMI and ARI) of the methods in the Speech domain.

Dataset	Metric	*Supervised Learning*							*Unsupervised*
		Proposed Methods			**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Isolet	*NMI*	0.834	**0.853**	0.796	0.818	0.812	0.747	0.829	0.816
	ARI	0.707	**0.733**	0.669	0.725	0.667	0.632	0.693	0.682
WSJ	*NMI*	0.813	**0.837**	0.811	0.831	0.810	0.521	0.707	0.555
	ARI	0.369	**0.385**	0.362	0.371	0.361	0.177	0.278	0.200
Average	*NMI*	24.35%	**27.67%**	21.84%	24.99%	22.73%	−7.29%	14.49%	
Improvement	*ARI*	44.08%	**49.98%**	39.55%	45.90%	39.15%	−9.42%	20.31%	

Figure 6.2 plots the improvement in NMI achieved by TCS-GM over baseline GM as we increase the number of training clusters from 3 to 25. Initially there is no improvement in performance, but as training clusters increase, there is a gradual improvement in performance, until it reaches a saturation level. This shows that (a) there is some minimum amount of supervision needed to see any improvement (b) providing more supervision does not increase performance indefinitely but saturates at certain level. This kind of curve is typical of most machine learning algorithms.

Table 6.7: The performance (in terms of NMI and ARI) of TCS-VM in the Text domain (with unit normalized data). For an informative comparison results of TCS-GM, TCS-CS are also tabulated.

Dataset	**Metric**	*Supervised Learning*							*Unsupervised*	
		Proposed Methods			**Existing Baselines**					
		TCS-VM	**TCS-GM**	**TCS-CS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**	**VM**
CNAE	*NMI*	**0.916**	0.815	0.803	0.204	0.692	0.614	0.693	0.792	0.909
	ARI	**0.95**	0.826	0.812	0.174	0.694	0.539	0.646	0.795	0.932
K9	*NMI*	**0.638**	0.621	0.620	0.351	0.479	0.617	0.584	0.616	0.615
	ARI	**0.514**	0.527	0.463	0.189	0.274	0.453	0.357	0.452	0.456
TDT4	*NMI*	**0.936**	0.916	0.915	0.895	0.923	0.914	0.911	0.915	0.933
	ARI	**0.871**	0.827	0.825	0.955	0.847	0.825	0.807	0.821	0.857
TDT5	*NMI*	**0.781**	0.750	0.756	0.760	0.707	0.750	0.756	0.755	0.766
	ARI	**0.393**	0.255	0.269	0.272	0.241	0.255	0.265	0.260	0.276

Table 6.8: The performance (in terms of NMI and ARI) of TCS-CS and GM in the presence of multiple valid clusterings. The percentage improvement of TCS-CS over GM is shown in parenthesis.

Metric	*Content-based Clustering*		*Speaker-based Clustering*	
	GM	**TCS-CS**	**GM**	**TCS-CS**
NMI	0.750	**0.773** (3.06%)	0.0373	**0.402** (986%)
ARI	0.527	**0.556** (5.50%)	0.0003	**0.238** ($\sim 80,000\%$)

6.8.2 Effect on Multiview Clustering

In this subsection we analyze how the supervision can help uncover multiple valid clusterings in the data. A typical unsupervised clustering algorithm will not know which view the user prefers, whereas our proposed TCS models should be able to align the instances to the user preferred view based on limited supervision.

We choose the *Isolet* dataset for this analysis because the dataset has two valid views - (a) clustering by content i.e. the alphabet uttered or (b) clustering by the speaker. There are a total of 26 alphabets and 150 speakers, 10% of which is set aside as supervision. As the results show (Table 6.8), the unsupervised GM model by default favors clusters by content and does a poor job when the user is interested in speaker based clustering. Our TCS-CS on the other hand, can succesfully detect which type of clustering is preferred and improve over GM by a significant margin.

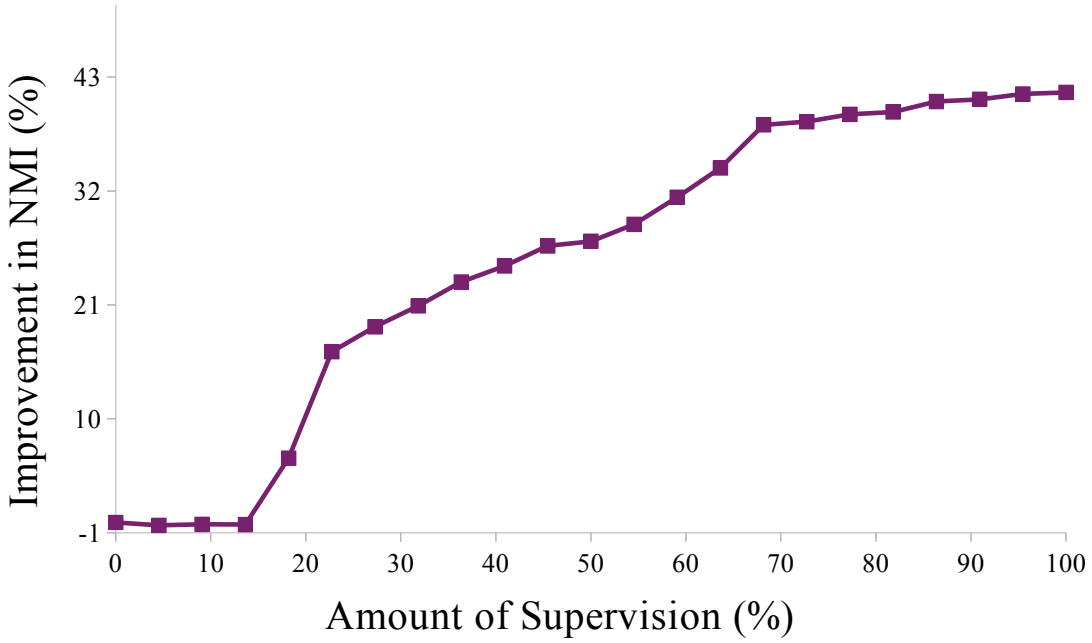


Figure 6.2: The Improvement in NMI on the test-set as the amount of training data (i.e. clusters) is increased on the WSJ dataset

6.8.3 Exploiting Unlabeled Data

Using available unlabeled data along with the labeled data has shown to improve supervised tasks such as label propagation [Zhu and Ghahramani, 2002], transductive SVMs [Joachims, 1999], certain regression tasks [Liang et al., 2007] etc. However, our setting is a little different in the sense that our final goal is not a supervised task, it is an unsupervised task of clustering the unlabeled data. Nevertheless, one can still try to use the unlabeled data while estimating the transformation function. Specifically, we employ a procedure similar to transductive SVM; first we use the labeled data to do an initial estimate of the transformation A, we then apply this transformation on the unlabeled data and cluster them. Now we assume that the new clusters in the unlabeled data are a part of ground-truth and re-estimate the transformation matrix. This can be iterated over multiple times and slowly increasing the contribution of the unlabeled examples in each iteration.

Let $\{\boldsymbol{X_L}, \boldsymbol{Y_L}\}$ and $\boldsymbol{X_U}$ denote the labeled and the unlabeled data respectively. $\boldsymbol{Y_U}$ denotes the cluster assignments of the unlabeled data and $\boldsymbol{C_L}, \boldsymbol{C_U}$ are the cluster parameters of the labeled and unlabeled data respectively. To estimate the unknowns $\{\boldsymbol{Y_U}, \boldsymbol{C_U}, \boldsymbol{C_L}\}$ we repeat

Table 6.9: The performance (in terms of NMI and ARI) of using just unlabeled data without supervision (GM), using supervision (TCS-GM) and using supervision with additional unlabeled data (TCS-GM-UL), on a few datasets. Using unlabeled data improves the performance.

Dataset	Metric	GM	TCS-GM	TCS-GM-UL
Aussign	*NMI*	0.470	0.591	**0.611**
	ARI	0.296	0.853	**0.858**
AT& T	*NMI*	0.803	0.803	**0.831**
	ARI	0.468	0.468	**0.549**
Vowel	*NMI*	0.274	0.309	**0.314**
	ARI	0.134	0.157	**0.163**
K9	*NMI*	0.523	0.559	**0.571**
	ARI	0.218	0.281	**0.518**

the following 3 steps starting with $\boldsymbol{\mu} = 0$,

STEP 1

$$\begin{aligned} \{L, \boldsymbol{C_L}\} \quad &= \arg\min_{L, \boldsymbol{C_L}} \quad \gamma\lambda(L) - \log P(\boldsymbol{Y_L}|L\boldsymbol{X_L}, \boldsymbol{C_L}) - \boldsymbol{\mu} \log P(\boldsymbol{Y_U}|L\boldsymbol{X_U}, \boldsymbol{C_U}) \\ &\text{s.t} \quad \boldsymbol{C_L} = \arg\max P(L\boldsymbol{X_L}|\boldsymbol{Y_L}, \boldsymbol{C_L}) \\ &\qquad\quad A \succeq 0 \end{aligned}$$

STEP 2

$$\{\boldsymbol{Y_U}, \boldsymbol{C_U}\} \quad = \arg\max_{\boldsymbol{Y_U}, \boldsymbol{C_U}} \quad P(L\boldsymbol{X_U}|\boldsymbol{Y_U}, \boldsymbol{C_U}) \quad \text{[clustering of unlabeled data]}$$

STEP 3

Increase μ slightly

The Y_U obtained is the cluster assignments of the unlabeled data. Preliminary analysis (Table 6.9) on a few datasets shows encouraging results, however a more thorough exploration is required to establish meaningful conclusions, especially with regards to the sensitivity to the parameter μ.

6.8.4 Effect of SVD on Clustering

One possible concern in reducing the dimension of the training data using SVD is that it could potentially lead to lower clustering performance. In order to understand the effect of SVD,

Table 6.10: Tabulates the results of using a Gaussian mixture clustering on the Full dataset (no dimension reduction) vs SVD based dimension reduced dataset.

Dataset	***Metric***	Full GM	SVD GM
Leaves	*NMI*	0.787	0.787
	ARI	0.438	0.438
Aussign	*NMI*	0.774	0.774
	ARI	0.400	0.408
Char	*NMI*	0.760	0.759
	ARI	0.593	0.600
DSPA	*NMI*	0.678	0.695
	ARI	0.298	0.320
Libras	*NMI*	0.578	0.581
	ARI	0.294	0.297
CNAE	*NMI*	0.389	0.389
	ARI	0.206	0.206
K9	*NMI*	0.542	0.548
	ARI	0.331	0.268
TDT4	*NMI*	0.854	0.885
	ARI	0.646	0.704
TDT5	*NMI*	0.834	0.797
	ARI	0.395	0.293

Dataset	***Metric***	Full GM	SVD GM
Penbased	*NMI*	0.684	0.684
	ARI	0.550	0.550
Letter	*NMI*	0.364	0.364
	ARI	0.133	0.133
USPS	*NMI*	0.800	0.793
	ARI	0.740	0.732
Binalpha	*NMI*	0.668	0.690
	ARI	0.378	0.422
Optrec	*NMI*	0.777	0.753
	ARI	0.713	0.685
AT & T	*NMI*	0.724	0.810
	ARI	0.357	0.504
Umist	*NMI*	0.646	0.647
	ARI	0.323	0.326
Faces96	*NMI*	0.873	0.842
	ARI	0.548	0.495
LFW	*NMI*	0.304	0.362
	ARI	0.009	0.009

we compare the performance of GM on the full dataset without dimension reduction against a SVD-based dimension reduced dataset. Table 6.10 tabulates the results by averaging over 10 different runs with Kmeans++ initialization.

On most datasets, there is no appreciable change in performance. In fact on some datasets - TDT4, AT & T, LFW there was a gain in performance by using SVD. There was a loss of performance only on two datasets - Faces96 and TDT5.

6.9 Summary

In this chapter we presented two ways of exploiting supervision to generate *better* clustering of unlabeled data. The first approach used a probabilistic framework that made parametric as-

sumptions about the distribution of the class conditional probabilities and the second approach used a non-probabilistic constrained optimization framework. Both our approaches rely on learning a transformation function that can capture the user expectations better. The transformation function is the key idea that helps us generalize the supervision in the form of observed clusters to discover similar unobserved clusters in the unlabeled data.

The results of our extensive testing on 23 datasets provide strong empirical support in favour of our proposed models against existing solutions and unsupervised baselines.

7 Future Work

There are several interesting directions for future research in each part of the thesis.

7.1 Large-Scale Classification with Structures

In chapters 2 and 3 we enforce similarity between parameters to model dependencies between class-labels. It would be interesting to see if more sophisticated ways of enforcing similarity between model parameters would be effective; for example,

1. In HBLR, one can incorporate hierarchical dependencies not just between the means of the Gaussians but also between the covariance matrices. One way to do this would be to have a hierarchical gamma prior for the variance terms (with a fixed shape parameter). This would try to capture how the variance of a parameter at a node is distributed among the children nodes.

2. In RR it would be interesting to learn a node specific linear transformation that is common to all the children nodes. One can envision enforcing a low rank structure as we go deeper into the hierarchy and a full rank structure at the higher levels of the hierarchy. It would be challenging to develop such models without loss in the scalability.

Another interesting direction would be to exploit the hierarchy or graph to subsample a node-specific training set. Since the relatedness of various class-labels is encoded in the hierarchy (or graph), we could develop ways to sub-sample the set of training data that is closest to the decision bounday of the particular class. For example the most naive way of doing this is the top down classifier which neglects all the training data except at sibling nodes. It would be interesting to see if there are more principled approaches that, for a given class, samples more

data from the confusable classes and less data from the less-confusable classes. The downside of the sampling is that is that some of the standard guarantees of risk minimization will be lost because the training sample is no longer from the true distribution. Some work along this direction has been done by [Gupta et al., 2014] where negative examples for a class are generated by first picking a class with equal probability followed by picking a random example.

7.2 Unsupervised Structure Extraction

In chapter 5 we developed our vMF models by assuming a single concentration parameter that determines the spread of the data. For low dimensional datasets, one can afford to have a full covariance structure between the parameters; this leads to the kent distribution [Mardia and Jupp, 2009]

$$f(x|\mu,\kappa,\Sigma) = \frac{1}{C(\kappa,\Sigma)} \exp\left(\kappa\mu^\top x + x^\top \Sigma x\right)$$

It would be intresting to extend the models and inference procedures to the kent distribution and see how modelling the covariance structures on unit-spheres improves the clustering results.

Another interesting direction would be to develop non-parametric Bayesian versions of the vMF models (useful when the number of clusters is not known apriori). This can be done by placing a dirichlet process prior on the means and the concentration parameter. The generative process is as follows,

$$\begin{aligned} G &\sim DP(.|G_0,\alpha) \\ \{\mu_i,\kappa_i\} &\sim G(.) \\ x_i &\sim \text{vMF}(.|\mu_i,\kappa_i) \end{aligned}$$

The base measure G_0 is the product space vMF$(.|\mu_0, C_0)$logNormal$(.|m,\sigma^2)$ and μ_0, C_0, m, σ^2 are the prior parameters that one needs to specify. One could perform MCMC sampling to estimate the posterior or use variational inference techniques by using the stick breaking process version of generating clusters. The extension to hierarchical clusters would be similar.

7.3 Semi-supervised Structure Expansion

7.3.1 Hierarchical Structures

It would be interesting to extend the TCS models to cases where there are hierarchical relations between classes in the supervision. More generally, given a partially labeled hierarchy as

supervision, the task is to complete the hierarchy based on the pool of unlabeled data.

The simplest way to do this would be apply either TCS or CSC (chapter 6) at each node of the hierarchy individually i.e. a node specific transformation matrix that all the children share. However this is not very effective because of two reasons - firstly, this would require some supervision under each node of the hierarchy (a few labeled nodes under each node) and secondly, we cannot use supervision to expand into an entirely new subtree, and to do so we must fallback to existing unsupervised hierarchical clustering methods.

Another way to expand the hierarchy is to enforce some structural dependencies between the transformation matrices at each node. For example, we can enforce that the parent and children transformation matrices must be close in most dimension except for a few; or that all the sibling transformation matrices have disjoint subspaces, etc. These structural dependencies can help exploit supervision even when creating an entirely new subtree.

7.3.2 Learning Transformation by Max-margin Formulation (TCS-MM)

Another way to learn a transformation function is by formulating an optimization problem similar to the SVM struct [Tsochantaridis et al., 2006]. Here the the target of interest is the cluster labels. The score of a set of labels is defined as the objective of the clustering algorithm when the instances are assigned to clusters based on the given labels. Denote $\mathcal{Y}$ as the space of all possible assignments of instances to clusters. Given one ground-truth clustering $y \in \mathcal{Y}$, we formulate the optimization problem as follows,

$$\begin{aligned} \min_{\mathbf{A}} \quad & \gamma\lambda(A) + C\xi \\ \text{s.t} \quad & F_A(y) \geq F_A(z) + \xi \quad \forall z \in \mathcal{Y}, z \neq y \\ & A \succeq 0 \end{aligned}$$

Here $F_A(z)$ denotes the clustering objective with labeling z. For example with simple Kmeans clustering,

$$F_A(z) = -\sum_{k=1}^{K} \sum_{i:z_i=k} (x_i - m_k)^\top A(x_i - m_k) \quad \text{where } k\text{'th cluster mean } m_k = \frac{1}{\sum_i I(z_i = k)} \sum_{i:z_i=k} x_i$$

In short, the constraints ensure that we estimate a transformation matrix such that ground-truth clustering has a better clustering objective than all possible clusterings by a margin ξ.

Typically such problems can be solved by using a constraint generation method where we rely on a subroutine (separation oracle) that is able to find the most violated constraint. Unfortunately, in our case finding the most violated constraint involves finding a z that maximizes $F_A(z)$ - which is the NP-hard clustering problem.

Table 7.1: The performance (in terms of NMI and ARI) of TCS-MM on all datasets.

Dataset	Metric	TCS-MM	Best other		Dataset	Metric	TCS-MM	Best other	
Aussign	*NMI*	0.864	**0.937**	(LMNN)	**CNAE**	*NMI*	**0.634**	0.608	(TCS-GM)
	ARI	0.630	**0.809**			*ARI*	**0.570**	0.574	
Char	*NMI*	0.728	**0.782**	(TCS-CS)	**K9**	*NMI*	0.557	**0.581**	(TCS-GM)
	ARI	0.609	**0.687**			*ARI*	0.359	**0.615**	
DSPA	*NMI*	0.702	**0.768**	(TCS-CS)	**TDT4**	*NMI*	0.902	**0.905**	(TCS-GM)
	ARI	0.407	**0.488**			*ARI*	0.817	**0.826**	
Libras	*NMI*	**0.675**	0.660	(TCS-CS)	**TDT5**	*NMI*	0.698	**0.699**	(TCS-GM)
	ARI	0.518	**0.537**			*ARI*	0.196	**0.198**	
Penbased	*NMI*	**0.656**	0.604	(TCS-GM	**AT& T**	*NMI*	0.836	**0.918**	(TCS-CS)
	ARI	**0.623**	0.504	-L2)		*ARI*	0.615	**0.758**	
Letter	*NMI*	0.512	**0.549**	(FD)	**Umist**	*NMI*	0.614	**0.826**	(FD)
	ARI	0.359	**0.385**			*ARI*	0.387	**0.676**	
USPS	*NMI*	0.822	**0.845**	(TCS-GM)	**Faces96**	*NMI*	0.910	**0.944**	(TCS-CS)
	ARI	0.792	**0.829**			*ARI*	0.709	**0.789**	
Binalpha	*NMI*	**0.801**	0.794	(TCS-GM)	**LFW**	*NMI*	**0.433**	0.415	(TCS-CS)
	ARI	**0.673**	0.669			*ARI*	**0.109**	0.095	
Optrec	*NMI*	0.725	**0.936**	(TCS-GM)	**Isolet**	*NMI*	**0.856**	0.855	(TCS-CS)
	ARI	0.733	**0.956**			*ARI*	**0.757**	0.733	
MNIST	*NMI*	0.651	**0.842**	(TCS-GM)	**WSJ**	*NMI*	0.810	**0.837**	(TCS-CS)
	ARI	0.857	**0.885**			*ARI*	0.359	**0.385**	

It would be interesting to see how an approximate constraint generator (i.e. generates *a* violated constraint instead of the *most* violated constraint) would work. We did some preliminary experiments where the constraints were generated using a simple clustering algorithm (initialized using kmeans++). The results for some of the datasets are shown in Table 7.1. Overall, the results mixed, sometimes there an impressive gain in performance for e.g. on LFW, CNAE, Isolet; whereas for most datasets there is not much more improvement than the other methods. The lack of a good constraint generator makes it difficult to understand the performance of the approach. It would be interesting to see if better methods can be designed for finding the most violated constraints for this problem.

Bibliography

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998. 69, 75

Kerem Altun and Billur Barshan. Human activity recognition using inertial/magnetic sensor units. In *Human Behavior Understanding*, pages 38–51. Springer, 2010. 92

A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. 2007. 26

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008. 26

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 89

Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, and Suvrit Sra. Generative model-based clustering of directional data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2003. 57

Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *The Journal of Machine Learning Research*, 6(2):1345, 2006. 57, 70

Mark Bangert, Philipp Hennig, and Uwe Oelfke. Using an infinite von mises-fisher mixture model to cluster treatment beam directions in external radiation therapy. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 746–751. IEEE, 2010. 57, 62

Árpád Baricz, Saminathan Ponnusamy, and Matti Vuorinen. Functional inequalities for modified bessel functions. *Expositiones Mathematicae*, 29(4):399–414, 2011. 62

S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *International Conference in Machine Learning, Workshop*, pages 19–26, 2002. 80, 81, 91

S. Basu, M. Bilenko, and R.J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004. 81

S. Basu, M. Bilenko, A. Banerjee, and R.J. Mooney. Probabilistic semi-supervised clustering with constraints. *Semi-supervised learning*, pages 71–98, 2006. 81

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. 86

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013. 49

Paul N Bennett and Nam Nguyen. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18. ACM, 2009. 2, 9

M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004. 81, 90

Christopher M Bishop and Michael E Tipping. Bayesian regression and classification. *Nato Science Series sub Series III Computer And Systems Sciences*, 190:267–288, 2003. 11

Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006. 16

David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006a. 4, 5, 47

David M Blei and John D Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007. 4, 70

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022, 2003. 4, 55, 70

David M Blei, T Griffiths, M Jordan, and J Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems*, 16:106–114, 2004. 4, 5, 70

MD Blei and JD Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems*, pages 147–155. Citeseer, 2006b. 4

John Blitzer, Kilian Q Weinberger, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, pages 1473–1480, 2005. 81

Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. URL http://leon.bottou.org/papers/bottou-98x. revised, oct 2012. 44

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August 2010. Springer. URL http://leon.bottou.org/papers/bottou-2010. 44

Guillaume Bouchard. Efficient bounds for the softmax function and applications to approximate inference in hybrid models. In *NIPS 2007 Workshop for Approximate Bayesian Inference in Continuous/Hybrid Systems*. Citeseer, 2007. 46, 47, 123

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 2011. 43, 49, 50, 51, 52

Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 86

Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87. ACM, 2004. 9, 26, 27

George Casella. Empirical bayes method - a tutorial. Technical report, Cornell University and Purdue University. 15, 63

George Casella. Empirical bayes gibbs sampling. *Biostatistics*, 2(4):485–500, 2001. 65

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7:31–54, 2006. 9, 26

Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992. 55

J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, 43(5):1470–1480, 1972. 44

Sanjoy Dasgupta. Learning mixtures of gaussians. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 634–644. IEEE, 1999. 88

H. Daumé III. Notes on cg and lm-bfgs optimization of logistic regression. 2004. 43

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007. 86

C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the International Conference on Music Information Retrieval*, pages 77–80, 2007. 9

Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning*, page 27. ACM, 2004. 9, 26

Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. Hierarchical annotation of medical images. *Pattern Recognition*, 44(10):2436–2449, 2011. 17, 22

Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29. ACM, 2004. 89

C.B. Do, C.S. Foo, and A.Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In *Neural Information Processing Systems*, volume 21, 2007. 10

Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56(1-3):9–33, 2004. 89

Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 2000. 9, 18

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004. 26

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pages 217–224. ACM, 2005. 82

Ronald Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 1953. 57

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936. 90, 91

Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989. 91

Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic press, 1990. 91

A. Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 2006. 16

Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *NIPS*, pages 513–520, 2004. 81, 90

S. Gopal and Y. Yang. Multilabel classification with meta-level features. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 315–322. ACM, 2010. 34

S. Gopal and Y. Yang. Transformation-based probabilistic clustering with supervision. In *Proceeding of the 30th conference on Uncertainty in Artificial Intelligence (UAI)*, 2014a. 99

Siddharth Gopal and Yiming Yang. Distributed training of large-scale logistic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 289–297, 2013a. 48

Siddharth Gopal and Yiming Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Special Interest Group in Knowledge Discovery and Datamining (KDD)*, 2013b. 27

Siddharth Gopal and Yiming Yang. Von mises-fisher clustering models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2014b. 72

Siddharth Gopal, Yiming Yang, and Alexandru Niculescu-Mizil. Regularization framework for large scale hierarchical classification. *Workshop in Large-scale Hierarchical Text Classification ECML*, 2012. 27

Nico Gornitz, Christian K Widmer, Georg Zeller, Andre Kahles, Gunnar Ratsch, and Soren Sonnenburg. Hierarchical multitask structured output learning for large-scale sequence segmentation. In *Advances in Neural Information Processing Systems*, pages 2690–2698, 2011. 9

Maya R Gupta, Samy Bengio, and Jason Weston. Training highly multiclass classifiers. *Journal of Machine Learning Research*, 15:1–48, 2014. 107

Peter Guttorp and Richard A Lockhart. Finding the location of a signal: A bayesian analysis. *Journal of the American Statistical Association*, 83(402):322–330, 1988. 57

Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009. 86

Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-Theory and Methods*, 6(9):813–827, 1977. 43

Kurt Hornik and Bettina Grün. movmf: An r package for fitting mixtures of von mises-fisher distributions. 71

C.J. Hsieh, K.W. Chang, C.J. Lin, S.S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML*, pages 408–415. ACM, 2008. 29, 32

K.L. Hsiung, S.J. Kim, and S. Boyd. Tractable approximate robust geometric programming. *Optimization and Engineering*, 9(2):95–118, 2008. 45

T. Jebara and A. Choromanska. Majorization for CRFs and latent likelihoods. In *Neural Information Processing Systems*, 2012. 123

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999. 102

Thorsten Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002. 4, 55

Armand Joulin, Francis Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1943–1950. IEEE, 2010. 82

Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010. 89

PE Jupp and KV Mardia. A unified view of the theory of directional statistics, 1975-1988. *International Statistical Review/Revue Internationale de Statistique*, 1989. 57

Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. 89

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. In *Proceedings of the 18th annual conference on Learning Theory*, pages 444–457. Springer-Verlag, 2005. 88

R.E. Kass and R. Natarajan. A default conjugate prior for variance components in generalized linear mixed models. *Bayesian Analysis*, 2006. 16

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. 1997. 9, 18

Andreas Krause, Pietro Perona, and Ryan G Gomes. Discriminative clustering by regularized information maximization. In *NIPS*, pages 775–783, 2010. 82

David D Lewis, Robert E Schapire, James P Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306. ACM, 1996. 127

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004. 33

Feng Liang, Sayan Mukherjee, and Mike West. The use of unlabeled data in predictive modeling. *Statistical Science*, pages 189–205, 2007. 102

C.J. Lin, R.C. Weng, and S.S. Keerthi. Trust region newton method for logistic regression. *The Journal of Machine Learning Research*, 9:627–650, 2008. 44

D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. 14, 43, 47, 51

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter*, 7(1):36–43, 2005. 2, 9, 18

Z. Lu and T. Leen. Semi-supervised learning with penalized probabilistic clustering. *Advances in neural information processing systems*, 17:849–856, 2005. 81

Zhengdong Lu and Miguel A Carreira-Perpinán. Constrained spectral clustering through affinity propagation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 82

Z.Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992. 15, 30

Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011. 85

Charles Mallah, James Cope, and James Orwell. Plant leaf classification using probabilistic integration of shape, texture and margin features, 2013. 96

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008. 17, 33, 93

Kanti V Mardia and Peter E Jupp. *Directional statistics*, volume 494. Wiley, 2009. 107

KV Mardia and El-Atoum. Bayesian inference for the vmf distribution. *Biometrika*, 1976. 57

B. Marlin, M.E. Khan, and K. Murphy. Piecewise bounds for estimating bernoulli-logistic latent gaussian models. In *Proceedings of the international conference on Machine learning*, 2011. 45

A. McCallum, R. Rosenfeld, T. Mitchell, and A.Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the international conference on Machine learning*, pages 359–367, 1998. 9

T.P. Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003. 44

Boaz Nadler and Meirav Galun. Fundamental limitations of spectral clustering. *NIPS*, 19:1017, 2007. 82

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. 49

Anthony O'Hagan, Jonathan Forster, and Maurice George Kendall. *Bayesian inference*. Arnold London, 2004. 4

Syama S Rangapuram and Matthias Hein. Constrained 1-spectral clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 1143–1151, 2012. 82

Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond Mooney. Spherical topic models. In *Proceedings of the international conference on Machine learning*, 2010. 57

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014. 49

Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004. 4, 55

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, 7:1601–1626, 2006. 9, 26

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986. 4, 55

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-newton method for online convex optimization. 2007. 43

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003. 43

B. Shahbaba and R.M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis*, 2(1):221–238, 2007. 10, 11, 19, 22, 25

D.F. Shanno. On broyden-fletcher-goldfarb-shanno method. *Journal of Optimization Theory and Applications*, 46(1):87–94, 1985. 43

Noam Shental, Tomer Hertz, Daphna Weinshall, and Misha Pavel. Adjustment learning and relevant component analysis. In *Computer VisionâĂŤECCV 2002*, pages 776–790. Springer, 2006. 90

A. Smola and R. Kondor. Kernels and regularization on graphs. *Learning theory and kernel machines*, pages 144–158, 2003. 26

Suvrit Sra. A short note on parameter approximation for von mises-fisher distributions: and a fast implementation of i s (x). *Computational Statistics*, 27(1):177–190, 2012. 64

Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000. 5

M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001. 11

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001. 30, 49

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *The Journal of Machine Learning Research*, 6(2):1453, 2006. 9, 18, 26, 27, 108

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 17th Annual International Conference on Machine Learning*, volume 1, pages 577–584, 2001. 80, 81

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008. 13, 59

Chong Wang, David Blei, and David Heckerman. Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*, 2012. 4

Xiang Wang and Ian Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–572. ACM, 2010. 82

Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006. 4

Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009. 89, 90

C. Widmer, J. Leiva, Y. Altun, and G. Rätsch. Leveraging sequence classification by taxonomy-based multitask learning. In *Research in Computational Molecular Biology*, pages 522–534. Springer, 2010. 9, 26

IPC WIPO. http://www.wipo.int/classifications /ipc/en/support/. 17

Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2002. 81

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *NIPS*, pages 1537–1544, 2004. 82

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 619–626. ACM, 2008. 9

Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 1794–1801. IEEE, 2009. 4

Yiming Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90, 1999. 127

Yiming Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145. ACM, 2001. 34

Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 96–103. ACM, 2003. 2, 9

H.F. Yu, F.L. Huang, and C.J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75, 2011. 44

Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. In *NIPS*, volume 1, pages 1057–1064, 2001. 89

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004. 44

Tong Zhang, Alexandrin Popescul, and Byron Dom. Linear prediction models with graph regularization for web-page categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–826. ACM, 2006. 26

D. Zhou, L. Xiao, and M. Wu. Hierarchical classification via orthogonal transfer. Technical report, MSR-TR-2011-54, 2011. 9, 18, 21, 27

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. 102

Variational Inference for Hierarchical Bayesian Logistic Regression

A

A.1 Variational Inference for model M2

For HBLR model discussed in section 2.4, the parameters of the approximate posterior are estimated by solving the following optimization problem

$$\max_{q(\mathbf{W},\boldsymbol{\alpha})\in\mathcal{Q}} \quad E_q\left[\log P(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] - E_q\left[\log q(\mathbf{W},\boldsymbol{\alpha})\right] \tag{A.1}$$

We use a coordinate ascent approach by maximizing the ELBO (A.1) w.r.t each posterior parameter. For example, to optimize the ELBO w.r.t w_n, we first identify the terms that depend on w_n and set the gradient to zero.

$$\text{ELBO}(w_n) = \int q(w_n|\mu_n,\Psi_n)\left[q(\mathbf{W}^{-w_n},\boldsymbol{\alpha})\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})d\mathbf{W}d\boldsymbol{\alpha}\right] - \int q(w_n|\mu_n,\Psi_n)\log q(w_n|\mu_n,\Psi_n)dw_n$$

where $\mathbf{W}^{-w_n}$ denotes all w except w_n. Setting the gradient w.r.t $q(w_n)$ to zero yields,

$$\begin{aligned}\log q^*(w_n|\mu_n,\Psi_n) &= E_{q^{-w_n}}\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] \;+\; constant \\ q^*(w_n|\mu_n,\Psi_n) &\propto \exp\left(E_{q^{-w_n}}\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right]\right)\end{aligned} \tag{A.2}$$

Similarly, to update α_n,

$$\begin{aligned}\log q^*(\alpha_n|\tau_n,\upsilon_n) &= E_{q^{-\alpha_n}}\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] \;+\; constant \\ q^*(\alpha_n|\tau_n,\upsilon_n) &\propto \exp\left(E_{q^{-\alpha_n}}\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right]\right)\end{aligned} \tag{A.3}$$

For both the parameters, the updates rely on the ability to calculate $E_q\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right]$ efficiently.

$$E_q[\log p(\boldsymbol{W},\boldsymbol{\alpha},\boldsymbol{D})] = E_q\left[\log P(\boldsymbol{D}|\boldsymbol{W},\boldsymbol{\alpha})\right] + E_q\left[\log P(\boldsymbol{W},\boldsymbol{\alpha})\right]$$

Where,

$$E_q\left[\log P(\boldsymbol{W},\boldsymbol{\alpha})\right] = \sum_{n\in\mathcal{N}} -\frac{1}{2}E_q\left[(w_n - w_{\pi(n)})^\top \Sigma^{-1}_{\pi(n)}(w_n - w_{\pi(n)})\right] - \sum_{n\in\mathcal{N}}\frac{1}{2}E_q\left[\log|\Sigma_n|\right] - \sum_{n\in\mathcal{N}/T}\sum_{i=1}^{d} E_q\left[\alpha_n^{(i)}\right] b_n + E_q\left[(a_n^{(i)}-1)\log(\alpha_n^{(i)})\right]$$

$$E_q\left[\log P(\boldsymbol{D}|\boldsymbol{W},\boldsymbol{\alpha})\right] = \sum_{(x,t)\in\mathbf{D}} E_q[w_n]^\top x - \sum_{(x,t)\in\mathbf{D}} E_q\left[\log\left(\sum_{n\in T}\exp(w_n^\top x)\right)\right] \tag{A.4}$$

The expectations of all the parameters except the log normalization in (A.4) are easy to compute. The log normalization constant involves the convolution of a normally distributed w_n with the soft-max function of w_n which has no closed form solution. The hardness of calculating this expectation has been well studied in previous literature [Bouchard, 2007], [Jebara and Choromanska, 2012]. We use the method proposed in [Bouchard, 2007] and lower bound the normalization constant with a quadratic function of w_n - a function whose expectation is easier to compute.

$$\begin{aligned}\log\left(\sum_{n\in T}\exp(w_n^\top x)\right) \leq& \beta_x + \sum_{n\in T}\frac{(w_n^\top x - \beta_x - \xi_{xn})}{2} + \sum_{n\in T}\log\left(1+\xi_{xn}^2\right)\\ &+\sum_{n\in T}\lambda(\xi_{xn})\left(\left(w_n^\top x-\beta_x\right)^2 - \xi_{xn}^2\right)\\ =& -(\frac{|T|}{2}-1)\beta_x + \sum_{n\in T}\frac{w_n^\top x}{2} - \sum_{n\in T}\frac{\xi_{xn}}{2} + \sum_{n\in T}\log(1+e^{\xi_{xn}})\\ &+\sum_{n\in T}\lambda(\xi_{xn})\left(w_n^\top(xx^\top)w_n - 2\beta_x(w_n^\top x)\lambda(\xi_{xn}) + \beta_x^2 - \xi_{xn}^2\right)\end{aligned} \tag{A.5}$$

where

$$\lambda(\xi_{xn}) = \frac{1}{2\xi_{xn}}\left(\frac{1}{1+\exp(-\xi_{xn})} - \frac{1}{2}\right)$$

Note that we have introduced variational parameter β_x and ξ_{xn} for every $x \in D, n \in \mathcal{N}$. These variation parameters are free parameters that we can optimize over to get the tightest possible bound. The expected log-sum-exp function can be computed as follows,

To compute the update for w_n (A.2)

$$E_{q^{-w_n}}\left[\log P(\mathbf{W},\boldsymbol{\alpha})\right] = E_{q^{-w_n}}\left[-\frac{1}{2}(w_n - w_{\pi(n)})^\top \Sigma^{-1}_{\pi(n)}(w_n - w_{\pi(n)})\right] + \sum_{c\in C_n} E_{q^{-w_n}}\left[-\frac{1}{2}(w_c - w_n)^\top \Sigma_n^{-1}(w_c - w_n)\right] + \text{constants..} \tag{A.6}$$

$$E_{q^{-w_n}}\left[\log P(\mathbf{D}|\mathbf{W},\boldsymbol{\alpha})\right] \geq \sum_{(x,t)\in\mathbf{D}} I(t=n) E_{q^{-w_n}}[w_n]^\top x - \sum_{(x,t)\in\mathbf{D}} \frac{w_n^\top x}{2} + \lambda(\xi_{xn})\left(w_n^\top \top x x^\top w_n - 2\beta_x w_n^\top x\right) + \text{constants..} \tag{A.7}$$

Putting together (A.6), (A.7) and (A.2), this implies that the mean μ_n and the covariance Ψ_n of the posterior normal distribution of w_n are given by,

$$\Psi_n^{-1} = I(n\in T)\sum_{(x,t)\in D} 2\lambda(\xi_{xn}) x x^\top + \operatorname{diag}(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}}) + |C_n|\operatorname{diag}(\frac{\tau_n}{\upsilon_n}) \tag{A.8}$$

$$\mu_n = \Psi_n\left(I(n\in T)\sum_{(x,t)\in D}(I(t=n) - \frac{1}{2} + 2\lambda(\xi_{xn})\beta_x)x + \operatorname{diag}(\frac{\tau_{\pi(n)}}{\upsilon_{\pi(n)}})\mu_{\pi(n)} + \operatorname{diag}(\frac{\tau_n}{\upsilon_n})\sum_{c\in C_n}\mu_c\right) \tag{A.9}$$

where we have used the fact that

$$\begin{aligned} E_{q^{-w_n}}[w_c] &= \mu_c \\ E_{q^{-w_n}}\left[\Sigma^{-1}_{n'}\right] &= \operatorname{diag}\left(E_{q^{-w_n}}\left[\alpha^{(1)}_{n'}\right], E_{q^{-w_n}}\left[\alpha^{(2)}_{n'}\right], \ldots, E_{q^{-w_n}}\left[\alpha^{(d)}_{n'}\right]\right) \\ &= \operatorname{diag}\left(\frac{\tau^{(1)}_{n'}}{\upsilon^{(1)}_{n'}}, \frac{\tau^{(2)}_{n'}}{\upsilon^{(2)}_{n'}}, \ldots, \frac{\tau^{(d)}_{n'}}{\upsilon^{(d)}_{n'}})\right) \qquad \forall n' \neq n \end{aligned}$$

For updating α_n,

$$\begin{aligned} E_{q^{-\alpha_n}}\left[\log p(\mathbf{W},\boldsymbol{\alpha},\mathbf{D})\right] &= -\sum_{c\in C_n}\frac{\alpha_n^{(i)}}{2}E_q\left[(w_n^{(i)} - w_c^{(i)})^2\right] + \left(\sum_{c\in C_n}\frac{\log(\alpha_n^{(i)})}{2}\right) - \alpha_n^{(i)} b_n^{(i)} + (a_n^{(i)} - 1)\log(\alpha_n^{(i)}) \\ &= -\alpha_n^{(i)}\left(b_n^{(i)} + \sum_{n\in C_n}\frac{1}{2}\left((\mu_c^{(i)} - \mu_n^{(i)})^2 + \Psi_n^{(i,i)} + \Psi_c^{(i,i)}\right)\right) \\ &\quad + (a_n^{(i)} + \frac{|C_n|}{2} - 1)\log(\alpha_n^{(i)}) + constant \end{aligned} \tag{A.10}$$

Combining equations (A.10), (A.3) implies that the posterior parameters of α_n are given by,

$$\upsilon_n^{(i)} = b_n^{(i)} + \sum_{c \in C_n} \Psi_n^{(i,i)} + \Psi_c^{(i,i)} + (\mu_n^{(i)} - \mu_c^{(i)})^2 \tag{A.11}$$

$$\tau_n^{(i)} = a_n^{(i)} + \frac{|C_n|}{2} \tag{A.12}$$

Note that at each step of optimization, the local variational parameters are also optimized to give the tightest possible bound. The variational parameters ξ_{xn}, β_x are updated by maximizing (A.5).

$$\xi_{xn}^2 = x^\top \operatorname{diag}(\frac{\tau_n}{\upsilon_n})x + (\beta_x - \mu_n^\top x)^2 \tag{A.13}$$

$$\beta_x = \frac{\left(\frac{1}{2}\left(\frac{1}{2}|T| - 1\right) + \sum_{n \in T} \lambda(\xi_{xn})\mu_n^\top x\right)}{\sum_{n \in T} \lambda(\xi_{xn})} \tag{A.14}$$

The full variational algorithm simply involes optimizing each of the variables as outlined in (A.8), (A.9), (A.11), (A.12), (A.13) and (A.14).

A.2 Variational Inference for model M1

Most of the derivation simply goes through changed except that $\alpha_n^{(1)} = \alpha_n^{(2)} = \ldots = \alpha_n$. The parameters are iteratively updated as follows,

$$\Psi_n^{-1} = \sum_{(x,t) \in D} 2\lambda(\xi_{xn})xx^\top + E_{q^{-w_n}}\left[\Sigma_{\pi(n)}^{-1}\right] + + |C_n| E_{q^{-w_n}}\left[\Sigma_n^{-1}\right]$$

$$\mu_n = \Psi_n \left(\sum_{(x,t) \in D} (I(t = n) - \frac{1}{2} + 2\lambda(\xi_{xn})\beta_x)x + E_{q^{-w_n}}\left[\Sigma_{\pi(n)}^{-1}\right]\mu_{\pi(n)} + E_{q^{-w_n}}\left[\Sigma_n^{-1}\right] \sum_{c \in C_n} \mu_c \right)$$

$$\upsilon_n = b_n + \sum_{c \in C_n} trace(\Psi_n) + trace(\Psi_c) + (\mu_n - \mu_c)^\top(\mu_n - \mu_c)$$

$$\tau_n = a_n + \frac{|C_n|d}{2}$$

where

$$E_{q^{-w_n}}\left[\Sigma_n^{-1}\right] = \operatorname{diag}\left(\frac{\tau_n}{\upsilon_n}, \frac{\tau_n}{\upsilon_n}, \ldots, \frac{\tau_n}{\upsilon_n})\right)$$

A.3 Variational Inference for model M3

The extension to HBLR-M3 follows along similar lines.

If $n \in T$

$$\Psi_n^{-1} = \sum_{(x,t)\in D} 2\lambda(\xi_{xn}) x x^\top + E_{q^{-w_n}}\left[\Sigma_n^{-1}\right]$$

$$\mu_n = \Psi_n \left(\sum_{(x,t)\in D} (I(t=n) - \frac{1}{2} + 2\lambda(\xi_{xn})\beta_x) x + E_{q^{-w_n}}\left[\Sigma_n^{-1}\right] \mu_{\pi(n)} \right)$$

If $n \notin T$

$$\Psi_n^{-1} = E_{q^{-w_n}}\left[\Sigma_n^{-1}\right] + \sum_{c\in C_n} E_{q^{-w_n}}\left[\Sigma_c^{-1}\right]$$

$$\mu_n = \Psi_n \left(E_{q^{-w_n}}\left[\Sigma_n^{-1}\right] \mu_{\pi(n)} + \sum_{c\in C_n} E_{q^{-w_n}}\left[\Sigma_c^{-1}\right] \mu_c \right)$$

$$\upsilon_n = b_n + trace(\Psi_{\pi(n)}) + trace(\Psi_n) + (\mu_n - \mu_{\pi(n)})^\top (\mu_n - \mu_{\pi(n)})$$

$$\tau_n = a_n + \frac{d}{2}$$

Evaluation Metrics

B

B.1 Classification Metrics

We use the most popular Micro-F_1 and Macro-F_1 evaluation measures.

1. **Micro-averaged** F_1 (Micro-F_1) is a conventional metric for evaluating classifiers [Lewis et al., 1996] [Yang, 1999]. The system-made decisions on test set with respect to a specific category $c \in \mathbf{C} \equiv \{c_1, ..., c_m\}$ can be divided into four groups: True Positives (TP_c), False Positives (FP_c), True Negatives (TN_c) and False Negatives (FN_c), respectively. The corresponding evaluation metrics are defined as:

$$P = \frac{\sum\limits_{c \in \mathbf{C}} TP_c}{\sum\limits_{c \in \mathbf{C}} (TP_c + FP_c)} \quad , \quad R = \frac{\sum\limits_{c \in \mathbf{C}} TP_c}{\sum\limits_{c \in \mathbf{C}} (TP_c + FN_c)} \quad , \quad \text{Micro-}F_1 = \frac{2PR}{P + R}$$

2. **Macro-averaged** F_1 (Macro-F_1) defined as:

$$P_c = \frac{TP_c}{TP_c + FP_c} \quad , \quad R_c = \frac{TP_c}{TP_c + FN_c} \quad , \quad \text{Macro-}F_1 = \frac{1}{m} \sum_{c \in \mathbf{C}} \frac{2P_c R_c}{P_c + R_c}$$

Both micro-averaged and macro-averaged are informative metrics. The former gives the performance on each instance an equal weight in computing the average; the latter gives the performance on each category an equal weight in computing the average.

B.2 Clustering Recoverability Metrics

The six clustering measures we used for ground-truth based evaluations are Normalized Mutual Information (**NMI**), Mutual Information (**MI**), Rand Index (**RI**), Adjusted Rand Index (**ARI**),

B.2. Clustering Recoverability Metrics

Purity and Macro-F_1. Following standard notation[1], given a collection of N instances, we let $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ denote predicted clusters and $C = \{c_1, c_2, ..c_J\}$ denote the true clusters.

1. **MI**: The standard definition of mutual information is

$$MI(\Omega, C) = I(\Omega, C) = \sum_{k,j} \frac{|\omega_k \cap C_j|}{N} \log\left(\frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}\right)$$

2. **NMI**: NMI is the normalized variant of MI, where we normalize MI by the sum of the entropies of the two random variables. This quantity always lies between 0 and 1, 0 indicating independence and 1 indicating complete dependence.

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{H(\Omega) + H(C)} \quad \text{where} \quad H(\Omega) = \sum_K \frac{|\omega_k|}{N} log(\frac{|\omega_k|}{N}) \ , \ H(C) = \sum_J \frac{|c_j|}{N} log(\frac{|c_j|}{N})$$

3. **RI**: We can view clustering as predicting a $\{0, 1\}$ decision for all $N(N-1)/2$ pairs of instances, where 1 indicates the pair of instances belong to the same cluster, and 0 indicates different clusters. Given the ground-truth decisions for the pairs, we can define the standard TP, TN, FP and FN for the predicted decisions. The rand index is defined as,

$$RI = \frac{TP + TN}{TP + TN + FP + FN}$$

4. **ARI**: ARI is the corrected-for-chance version of the Rand index. Define $n_{j.} = \sum_k |c_j \cap \omega_k|$ and $n_{.k} = \sum_j |c_k \cap \omega_k|$, and $n_{ij} = |c_k \cap \omega_k|$.

$$ARI = \frac{\sum_{k,j} \binom{n_{ij}}{2} - \left[\sum_j \binom{n_{j.}}{2} \sum_k \binom{n_{.k}}{2}\right] / \binom{N}{2}}{\frac{1}{2}\left[\sum_j \binom{n_{j.}}{2} + \sum_k \binom{n_{.k}}{2}\right] - \left[\sum_j \binom{n_{j.}}{2} \sum_k \binom{n_{.k}}{2}\right] / \binom{N}{2}}$$

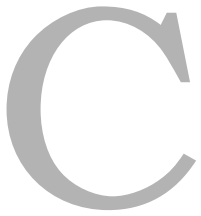
5. **Purity**: Each cluster is assigned to the class which is most frequent in the cluster, and purity is the accuracy of this assignment,

$$Purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

6. **Macro-F_1**: Each cluster is assigned to the class which is most frequent in the cluster. Now we can compute any classification measure including accuracy (Purity) or Macro-F_1.

[1] http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

Learning Transformation by CPM



C.1 General Mixture Model

We discuss how to estimate a generic transformation $G(x)$ for $x \in \mathcal{X}$ given a mixture of K densities $f(x|C_k)$ over $\mathcal{X}$, where C_k denotes the parameters of cluster k ($k \in \{1, 2, ..K\}$). We assume that we are given supervised information $\mathcal{S} = \{x_i, t_i\}_{i=1}^N$ where $x_i \in \mathcal{X}$ and $t_i \in \{1, 2..K\}$. For convenience define $y_{ik} = I(t_i = k)$. Note that the transformation $G(x)$ needs to appropriately defined such that the transformation still lies within the support of the distribution.

The conditional distribution for class k given an instance x is defined as, (note that k is also overloaded to represent class k)

$$P(k|\mathbf{C}, x_i) = \frac{f(x|C_k)}{\sum_{j=1}^{K} f(x|C_j)}$$

The transformation G is estimated by maximizing the conditional distribution of the labels given the transformed instances $G(\mathbf{X})$.

$$\log P(\mathbf{Y}|G(\mathbf{X}), \mathbf{C}^{mle}(G)) = \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\left[\log f(G(x_i)|C_k^{mle}(G)) - \log\left(\sum_{j=1}^{K} f(G(x_i)|C_j^{mle}(G))\right)\right]$$

Here $C_k^{mle}(G)$ denotes the MLE estimate of the k'th cluster parameter under the transformation G.

Since we doing supervised learning, there is a good chance that the transformation will overfit the training data. We therefore add a regularizer $\lambda(G)$ with a regularization constant γ.

Putting everything together, the optimization problem to solve is,

$$\max_{G,\mathbf{C}} \quad \gamma\lambda(G) + \sum_{i=1}^{N}\left[\sum_{k=1}^{K} y_{ik}\log f(G(x_i)|C_k)\right] - \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K} f(G(x_i)|C_k)\right)$$
$$\text{s.t.} \quad C_k = \arg\max_{C'_k}\sum_{i=1}^{N} y_{ik}\log f(G(x_i)|C'_k) \qquad \text{(i.e. } C \text{ is the MLE estimate)}$$

The argmax constraint can be succintly rewritten as equalitys for each C_k constraint. Rewriting as a minimization problem,

$$\min_{G,\mathbf{C}} \quad \gamma\lambda(G) - \sum_{i=1}^{N}\left[\sum_{k=1}^{K} y_{ik}\log f(G(x_i)|C_k)\right] + \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K} f(G(x_i)|C_k)\right) \tag{C.1}$$
$$\text{s.t.} \quad \frac{\partial\sum_{i=1}^{N} y_{ik}\log f(G(x_i)|C'_k)}{\partial C_k} = 0 \tag{C.2}$$

For arbitrary distributions, there is no general recipe to solve this problem. However for certain distributions, closed form expressions to the equality constraint are available, which can substituted directly into the minimization problem (C.1). Sometimes this can lead to a convex optimization problem (for e.g. in the case of Gaussian distribution).

Note at the optimal solution $\mathbf{C}^{mle}$, the gradient w.r.t G satisfies the following,

$$\gamma\lambda'(G) + \sum_{i=1}^{N}\sum_{k=1}^{K}\left[y_{ik} - P(k|C_k^{mle}, G(x_i))\right]\frac{\partial\log f(G(x_i)|C_k^{mle})}{\partial G} = 0$$

C.2 Gamma Mixtures

Gamma distribution is defined over $\mathcal{X} \equiv \mathcal{R}_+$. We consider a multivariate version of gamma distribution where each dimension is independently drawn. In a P dimensional space, the density for a point $x \in \mathcal{R}_+^P$ is given by,

$$P(x_i|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{p=1}^{P}\frac{\beta_p^{\alpha_p}}{\Gamma(\alpha_p)}x_{ip}^{\alpha_p-1}e^{-\beta_p x_{ip}}$$

We consider a mixture of gamma distributions where each cluster is associated with cluster parameter $\{\alpha_k,\beta_k\}_{k=1}^K$ where $\alpha_k,\beta_k \in \mathcal{R}_+^P$. We define a transformation,

$$G(x_i) = Lx_i \quad , \quad L \in \mathcal{R}_+^{P\times P}$$

The log probability of the transformed point Lx_i from cluster k is given by,

$$\log P(Lx_i|k) = \sum_{p=1}^{P} \alpha_k^p \log(\beta_k^p) - \sum_{p=1}^{P} \log\Gamma(\alpha_k^p) + \sum_{p}^{P}(\alpha_k^p - 1)\log(L_p^T x_i) - \sum_{p=1}^{P} \beta_k^p (L_p^T x_i)$$

where L_p is row p in L

Note that the MLE constraint for α_k, β_k are given by (m_k denotes the mean of all instances assigned to cluster k)

$$\beta_k^p = \frac{\alpha_k^p}{L_p^\top m_k}$$

$$\log(\beta_k^p) - \psi(\beta_k^p) = \log(L_p^\top m_k) - \frac{1}{n_k}\sum_{i=1}^{N}\log(L_p^\top x_i)$$

Since the MLE estimate of β_k^p is relatively simple, we substitute it into the optimization problem (**??**) with the gamma density defined above. We follow a similar strategy like vMF where alternatively optimize α_k^p and L. In the first step, we find the α_k^p that satisfies the MLE constraint using newton's method, and in the second step we fix the α_k's and optimize for L.

The derivative of above w.r.t row L_p is given by,

$$\sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - p_{ik})\frac{\partial \log P(Lx_i|\alpha^k, \beta^k)}{\partial L_p}$$

Considering summation with just class K, the derivative w.r.t L_p (the pth row) is given by,

$$\begin{aligned}\text{Gradient } = &\left(\frac{\alpha_k^p m_k}{(L_p^\top m_k)}\sum_{i=1}(y_{ik} - P(k|Lx_i))\right) + \left((\alpha_k^p - 1)\sum_{i=1}^{N}(y_{ik} - P(k|Lx_i))\frac{1}{L_p^\top x_i}x_i\right) \\ &- \left(\frac{\alpha_k^p}{L_p^\top m_k}\sum_{i=1}^{N}(y_{ik} - p(k|Lx_i))x_i\right) + \left(\frac{m_k}{(L_p^\top m_k)^2}\sum_{i=1}^{N}(y_{ik} - P(k|Lx_i))L_p^\top x_i\right)\end{aligned}$$

This can be rewritten as,

$$\text{Define} \quad n_k = \sum_{i=1}^{N} y_{ik} \quad , \quad z_k = \sum_{i=1}^{N} P(k|Lx_i) \quad , \quad J_{ikp} = \frac{(y_{ik} - p(k|Lx_i))}{L_p^\top x_i}$$

$$J_{kp} = \sum_{i=1}^{N} J_{ikp}x_i \quad , \quad s_k = \sum_{i=1}^{N}(y_{ik} - p(k|Lx_i))x_i$$

$$\text{Gradient} = \left(\frac{\alpha_k^p}{L_p^\top m_k}(n_k - z_k)m_k\right) + (\alpha_k^p - 1)J_{kp} - \left(\frac{\alpha_k^p}{L_p^\top m_k}s_k\right) + \left(\frac{L_p^\top s_k}{(L_p^\top m_k)^2}m_k\right)$$

We follow a similar procedure like vMF models. Note that things can simply a lot if we assume that the shape parameter is known or can be estimated from all the data initially.

Detailed Results of TCS Models

D

D.1 Comparison of TCS models with Baselines

The following tables all the complete results of all the models on all the datasets using 5 clustering recoverability metrics. We also include the results of additional **TCS-FS** method which is a small tweak to Fisher discriminant analysis **FD**. TCS-FS is essentially the same as **FD** except the the S_B (in eq (6.4)) is defined slightly differently as below,

$$S_B = \sum_{i=1}^{K} \sum_{j=i+1}^{K} (m_i - m_j)(m_i - m_j)^\top$$

Here we try to find a subspace where the distance between every pair of clusters is maximized and within cluster distances are minimized. Unliked **FD**, we completely remove the effect of any within cluster similarity in the computation of S_B matrix.

Table D.1: The performance of the methods on other datasets.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Image	*NMI*	**0.836**	0.836	0.399	0.375	0.369	0.495	0.601	0.699	0.780
	MI	**0.915**	0.914	0.427	0.406	0.399	0.529	0.640	0.768	0.851
	ARI	**0.839**	0.828	0.344	0.323	0.314	0.421	0.533	0.697	0.794
	RI	**0.928**	0.923	0.700	0.694	0.690	0.735	0.785	0.865	0.906
	Purity	**0.939**	0.937	0.715	0.699	0.691	0.727	0.720	0.878	0.908
Vowel	*NMI*	**0.414**	0.403	0.394	0.401	0.361	0.346	0.135	0.235	0.253
	MI	**0.662**	0.556	0.624	0.641	0.580	0.546	0.212	0.370	0.400
	ARI	0.273	0.287	0.274	**0.300**	0.265	0.244	0.066	0.159	0.149
	RI	0.766	0.764	0.760	**0.774**	0.766	0.749	0.686	0.718	0.720
	Purity	0.521	0.536	0.522	**0.556**	0.534	0.479	0.325	0.450	0.396
Leaves	*NMI*	0.824	0.876	0.822	**0.903**	0.902	0.852	0.767	0.815	0.778
	MI	2.855	3.036	2.850	**3.115**	3.114	2.951	2.650	2.802	2.687
	ARI	0.592	0.679	0.590	**0.710**	0.709	0.637	0.491	0.559	0.512
	RI	0.975	0.980	0.975	0.982	**0.982**	0.978	0.968	0.972	0.970
	Purity	0.725	0.783	0.725	0.795	**0.796**	0.751	0.650	0.702	0.665

Table D.2: The performance of the methods in the Time-series domain.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Aussign	*NMI*	0.890	0.910	0.925	0.932	0.931	**0.937**	0.775	0.838	0.817
	MI	3.048	3.123	3.173	3.191	3.189	**3.229**	2.634	2.856	2.780
	ARI	0.694	0.742	0.764	0.768	0.768	**0.809**	0.473	0.588	0.546
	RI	0.980	0.983	0.985	0.985	0.985	**0.988**	0.963	0.972	0.968
	Purity	0.771	0.809	0.814	0.822	0.819	**0.852**	0.609	0.694	0.668
Char	*NMI*	0.752	**0.782**	0.754	0.645	0.668	0.671	0.690	0.737	0.687
	MI	1.532	**1.592**	1.540	1.323	1.370	1.375	1.409	1.507	1.405
	ARI	0.648	**0.687**	0.652	0.563	0.505	0.572	0.588	0.622	0.568
	RI	0.917	**0.926**	0.919	0.900	0.889	0.902	0.904	0.913	0.900
	Purity	0.761	**0.781**	0.773	0.739	0.701	0.728	0.729	0.750	0.721
DSPA	*NMI*	0.685	**0.768**	0.734	0.734	0.739	0.597	0.668	0.642	0.678
	MI	1.231	**1.433**	1.367	1.323	1.335	0.968	1.204	1.124	1.219
	ARI	0.406	0.488	**0.518**	0.319	0.468	0.253	0.392	0.340	0.398
	RI	0.825	0.835	**0.873**	0.743	0.842	0.668	0.822	0.785	0.823
	Purity	0.660	**0.743**	0.712	0.579	0.693	0.519	0.654	0.605	0.659
Libras	*NMI*	0.608	0.660	0.642	**0.664**	0.649	0.498	0.599	0.592	0.512
	MI	0.945	**1.024**	1.010	1.019	1.014	0.769	0.939	0.918	0.803
	ARI	0.474	0.537	**0.540**	0.516	0.522	0.375	0.474	0.454	0.376
	RI	0.821	0.843	**0.849**	0.830	0.838	0.781	0.825	0.814	0.794
	Purity	0.693	**0.737**	0.736	0.726	0.731	0.623	0.687	0.681	0.631

Table D.3: The performance of the methods in the Text domain.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
CNAE	*NMI*	**0.608**	0.528	0.239	0.435	0.401	0.314	0.431	0.481	0.485
	MI	**0.638**	0.530	0.246	0.443	0.439	0.309	0.418	0.476	0.480
	ARI	**0.574**	0.405	0.202	0.373	0.384	0.194	0.285	0.355	0.359
	RI	**0.800**	0.703	0.626	0.712	0.726	0.598	0.630	0.674	0.676
	Purity	**0.807**	0.720	0.566	0.714	0.675	0.611	0.649	0.684	0.684
K9	*NMI*	**0.581**	0.562	0.405	0.335	0.410	0.375	0.562	0.510	0.561
	MI	**0.952**	0.894	0.654	0.552	0.704	0.605	0.901	0.780	0.889
	ARI	**0.615**	0.384	0.456	0.097	0.185	0.178	0.367	0.259	0.365
	RI	**0.853**	0.761	0.788	0.682	0.713	0.693	0.756	0.696	0.753
	Purity	**0.768**	0.728	0.670	0.557	0.641	0.625	0.731	0.699	0.733
TDT4	*NMI*	**0.905**	0.902	0.689	0.898	0.892	0.900	0.891	0.894	0.898
	MI	1.948	1.956	1.498	**1.971**	1.956	1.951	1.921	1.939	1.937
	ARI	**0.826**	0.824	0.441	0.824	0.808	0.807	0.794	0.798	0.809
	RI	0.959	0.959	0.869	**0.960**	0.956	0.954	0.949	0.952	0.954
	Purity	**0.926**	0.924	0.735	0.920	0.920	0.923	0.907	0.915	0.914
TDT5	*NMI*	**0.699**	0.696	0.695	0.699	0.697	0.675	0.692	0.694	0.692
	MI	**2.159**	2.150	2.146	2.156	2.149	2.085	2.135	2.127	2.135
	ARI	0.198	**0.200**	0.199	0.197	0.198	0.192	0.193	0.196	0.194
	RI	0.855	**0.856**	0.855	0.855	0.855	0.854	0.854	0.853	0.854
	Purity	**0.862**	0.857	0.859	0.862	0.862	0.837	0.856	0.857	0.858

D.1. Comparison of TCS models with Baselines

Table D.4: The performance of the methods in the Handwritten Characters Domain.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Penbased	*NMI*	0.556	0.572	**0.604**	0.602	0.464	0.571	0.401	0.487	0.522
	MI	0.753	0.775	0.796	**0.820**	0.625	0.769	0.550	0.658	0.707
	ARI	0.509	0.501	0.504	**0.562**	0.395	0.531	0.362	0.417	0.454
	RI	0.809	0.802	0.807	**0.831**	0.762	0.814	0.757	0.771	0.786
	Purity	0.736	0.715	0.736	**0.777**	0.639	0.758	0.649	0.663	0.686
Letter	*NMI*	0.478	0.484	0.504	0.540	**0.549**	0.434	0.265	0.373	0.351
	MI	1.078	1.081	1.135	1.211	**1.229**	0.976	0.596	0.841	0.792
	ARI	0.301	0.301	0.329	0.369	**0.385**	0.245	0.116	0.193	0.175
	RI	0.867	0.861	0.867	0.877	**0.878**	0.854	0.828	0.843	0.841
	Purity	0.501	0.497	0.525	0.550	**0.554**	0.459	0.307	0.390	0.369
USPS	*NMI*	**0.845**	0.810	0.455	0.637	0.444	0.785	0.807	0.786	0.815
	MI	**1.129**	1.084	0.594	0.861	0.604	1.050	1.075	1.043	1.088
	ARI	**0.829**	0.776	0.427	0.628	0.380	0.757	0.776	0.725	0.784
	RI	**0.932**	0.911	0.763	0.854	0.759	0.904	0.909	0.885	0.913
	Purity	**0.916**	0.888	0.682	0.811	0.644	0.882	0.888	0.863	0.893
Binalpha	*NMI*	**0.794**	0.755	0.703	0.731	0.734	0.703	0.725	0.680	0.719
	MI	**1.948**	1.856	1.723	1.794	1.804	1.732	1.778	1.672	1.765
	ARI	**0.669**	0.607	0.550	0.595	0.599	0.567	0.559	0.509	0.545
	RI	**0.947**	0.938	0.928	0.936	0.937	0.931	0.930	0.923	0.928
	Purity	**0.795**	0.742	0.715	0.754	0.741	0.731	0.682	0.666	0.677
Optrec	*NMI*	**0.936**	0.727	0.727	0.410	0.392	0.905	0.864	0.914	0.912
	MI	**1.028**	1.001	0.799	0.567	0.543	0.995	0.947	1.002	0.998
	ARI	**0.956**	0.719	0.791	0.391	0.377	0.939	0.880	0.929	0.924
	RI	**0.981**	0.893	0.907	0.772	0.765	0.973	0.946	0.968	0.965
	Purity	**0.985**	0.872	0.926	0.669	0.667	0.979	0.955	0.973	0.968
MNIST	*NMI*	**0.842**	0.832	0.701	0.549	0.366	0.717	0.553	0.741	0.830
	MI	**1.165**	1.152	0.968	0.760	0.506	0.983	0.761	1.016	1.149
	ARI	**0.885**	0.877	0.719	0.567	0.276	0.724	0.508	0.740	0.875
	RI	**0.957**	0.954	0.894	0.837	0.728	0.893	0.813	0.899	0.953
	Purity	**0.955**	0.951	0.883	0.793	0.523	0.860	0.731	0.877	0.950

Table D.5: The performance of the methods on Face clustering datasets.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
AT & T	*NMI*	0.843	**0.918**	0.879	0.892	0.862	0.842	0.822	0.852	0.834
	MI	2.173	**2.371**	2.266	2.315	2.239	2.182	2.119	2.197	2.151
	ARI	0.627	**0.758**	0.692	0.736	0.692	0.652	0.590	0.640	0.613
	RI	0.948	**0.966**	0.957	0.965	0.959	0.953	0.943	0.950	0.946
	Purity	0.755	**0.847**	0.794	0.842	0.813	0.784	0.726	0.772	0.746
Umist	*NMI*	0.588	0.806	0.739	**0.848**	0.826	0.792	0.569	0.563	0.554
	MI	1.180	1.603	1.491	**1.715**	1.681	1.574	1.153	1.129	1.112
	ARI	0.355	0.593	0.548	**0.685**	0.676	0.600	0.355	0.332	0.322
	RI	0.843	0.893	0.894	0.922	**0.924**	0.895	0.849	0.838	0.836
	Purity	0.579	0.735	0.712	**0.804**	0.751	0.746	0.560	0.557	0.552
Faces96	*NMI*	0.922	**0.944**	0.929	0.942	0.942	0.939	0.887	0.894	0.886
	MI	3.568	**3.662**	3.600	3.654	3.654	3.642	3.438	3.465	3.438
	ARI	0.728	**0.789**	0.752	0.783	0.783	0.771	0.662	0.673	0.663
	RI	0.989	**0.991**	0.990	0.991	0.991	0.991	0.986	0.987	0.986
	Purity	0.794	**0.839**	0.810	0.836	0.836	0.834	0.747	0.756	0.745
LFW	*NMI*	0.388	0.415	**0.415**	0.413	0.414	0.331	0.312	0.297	0.285
	MI	1.456	1.556	**1.557**	1.550	1.555	1.239	1.168	1.115	1.070
	ARI	0.081	0.089	**0.095**	0.089	0.089	0.040	0.030	0.026	0.021
	RI	0.939	0.940	**0.940**	0.940	0.940	0.936	0.936	0.936	0.935
	Purity	0.341	0.364	0.367	**0.373**	0.372	0.271	0.253	0.246	0.233

Table D.6: The performance of the methods in the Speech domain.

Dataset	Metric	*Supervised Learning*								*Unsupervised*
		Proposed Methods				**Existing Baselines**				
		TCS-GM	**TCS-CS**	**TCS-GM-L2**	**TCS-FS**	**FD**	**LMNN**	**PCC**	**BP**	**GM**
Isolet	*NMI*	0.834	**0.853**	0.796	0.812	0.818	0.812	0.747	0.829	0.816
	MI	1.878	**1.922**	1.803	1.843	1.861	1.831	1.700	1.874	1.846
	ARI	0.707	**0.733**	0.669	0.714	0.725	0.667	0.632	0.693	0.682
	RI	0.942	0.946	0.937	0.945	**0.947**	0.934	0.931	0.940	0.938
	Purity	0.813	**0.837**	0.798	0.824	0.829	0.761	0.748	0.786	0.767
WSJ	*NMI*	0.813	0.831	0.811	**0.837**	0.803	0.810	0.521	0.707	0.555
	MI	2.678	2.740	2.674	**2.753**	2.651	2.672	1.742	2.356	1.856
	ARI	0.369	0.371	0.362	**0.385**	0.348	0.361	0.177	0.278	0.200
	RI	0.914	0.914	0.913	**0.915**	0.911	0.913	0.891	0.904	0.894
	Purity	0.854	0.865	0.852	**0.870**	0.849	0.854	0.610	0.785	0.647